mathematics

MDPI

Article

A Multi-Objective Particle Swarm Optimization Algorithm Based on Gaussian Mutation and an Improved Learning Strategy

Ying Sun [1,†] and Yuelin Gao [1,2,*,†]

[1] School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China; sunying@nun.edu.cn

[2] Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China

* Correspondence: 1993001@nun.edu.cn; Tel.: +86-951-206-6579

† These authors contributed equally to this work.

Received: 8 January 2019; Accepted: 29 January 2019; Published: 4 February 2019

Abstract: Obtaining high convergence and uniform distributions remains a major challenge in most metaheuristic multi-objective optimization problems. In this article, a novel multi-objective particle swarm optimization (PSO) algorithm is proposed based on Gaussian mutation and an improved learning strategy. The approach adopts a Gaussian mutation strategy to improve the uniformity of external archives and current populations. To improve the global optimal solution, different learning strategies are proposed for non-dominated and dominated solutions. An indicator is presented to measure the distribution width of the non-dominated solution set, which is produced by various algorithms. Experiments were performed using eight benchmark test functions. The results illustrate that the multi-objective improved PSO algorithm (MOIPSO) yields better convergence and distributions than the other two algorithms, and the distance width indicator is reasonable and effective.

Keywords: multi-objective optimization problems; particle swarm optimization (PSO); Gaussian mutation; improved learning strategy

1. Introduction

Multi-objective optimization problems (MOPs) are very common in engineering and other areas of research, such as economics, finance, production scheduling, and aerospace engineering. It is very difficult to solve these problems because they usually involve several conflicting objectives. Generally, the optimal solution of MOPs is a set of optimal solutions (known as a Pareto optimal set), which differs from the solution of single-objective optimization (with only one optimal solution) [1]. Some classical optimization methods (weighted methods, goal programming methods, etc.) require the problem functions to be differentiable and are required to run multiple times with the hope of finding different solutions. In recent years, the multi-objective optimization evolutionary algorithm (MOEA) has become a popular method for solving MOPs, and it has garnered scholarly interest around the world [2]. Many representative MOEAs, such as multiple objective genetic algorithm (MOGA) [3], non-dominated sorting genetic algorithm II (NSGA-II) [4], strength pareto evolutionary algorithm (SPEA) [5] and pareto archived evolution strategy (PAES) [6], have been presented.

Over the past decade, the particle swarm optimization algorithm (PSO) has been used to solve MOPs, and a number of multi-objective PSO algorithms have been suggested. Some studies have shown that the modified PSO algorithm can effectively solve the MOPs and that the non-dominated solution set of the algorithm is much closer to the true Pareto optimal front. Coello and Pultdo [7]

incorporated a special mutation operator that enriches the exploratory capabilities. Sierra and Coello [8] proposed the OMOPSO algorithm, which places the whole population into three subpopulations with the same size and uses different mutation operators within different subpopulations. This algorithm improves the exploration ability of the particles. Reddy and Kumar [9] proposed an elitist-mutation multi-objective PSO (EM-MOPSO) algorithm with a strategic mechanism that effectively explores the feasible search space and speeds up the search for the true Pareto optimal region. Leong and Yen [10] proposed a dynamic population multiple-swarm MOPSO algorithm that uses an adaptive local archive to improve the diversity within each swarm. Yen and Leong [11] also proposed a dynamic population multiple-swarm MOPSO algorithm in which the number of swarms is adaptively adjusted throughout the search process via the proposed dynamic swarm strategy. The strategy allocates an appropriate number of swarms to support convergence and diversity criteria among the swarms as required. Chen, Zou, and Wang [12] presented a multi-objective endocrine particle swarm optimization algorithm (MOEPSO) in which the hormone (RH), released by the endocrine system, is encoded as a particle swarm and is then supervised by the corresponding stimulating hormone. Lin et al. [13] introduced a novel MOPSO algorithm using multiple search strategies (MMOPSO), and a decomposition approach was used to transform MOPs into a set of aggregation problems. Then, each particle was accordingly assigned to optimize each aggregation problem. A multi-objective vortex particle swarm optimization (MOVPSO) method was proposed in [14] based on the emerging properties of a swarm to simulate motion with diversity control via collaborative mechanisms using linear and circular movements. The parallel cell coordinate system (PCCS) in self-adaptive MOPSO [15] is used to select global and personal bests, maintain archives and adjust flight parameters. Knight et al. [16] presented a spreading mechanism to promote diversity in MOPSO. Cheng et al. [17] presented a hybrid multi-objective particle swarm optimization that combines the canonical PSO search with a teaching–learning-based optimization (TLBO) algorithm to promote diversity and improve the search ability. Overall, for any improved MOPSO algorithm, the search ability is determined by the neighbouring topological structure, and the convergence rate depends on the dominance relationship.

In this paper, we introduce a new multi-objective PSO algorithm based on Gaussian mutation and an improved learning strategy to solve MOPs. The main new contributions of this work can be summarized as: (1) Gaussian mutation throw points strategy to improve the uniformity of external archives and current populations; (2) For MOPs, it is difficult to select the g_{best} value of velocity and update the formula. Unlike other MOPSOs, that often randomly select a solution from the external archive as the global optimal solution g_{best}, we present different learning strategies to update the individual positions of the non-dominated and dominated solutions; (3) To further measure the distribution width, the indicator DW is proposed.

The remainder of this paper is organized as follows. In Section 2, we describe the multi-objective optimization. Thereafter, in Section 3, we present a multi-objective improved PSO algorithm (MOIPSO). Section 4 outlines the MOIPSO algorithm. Test problems, performance measures, and the results are provided in Section 5, and the conclusions are presented in Section 6.

2. Description of Multi-Objective Optimization Problems

A general minimization problem of m objectives can be mathematically stated as follows:

Given $x = (x_1, x_2, \cdots, x_n) \in D$, $D \subset R^n$ where n is the dimension of decision variable space D. Additionally,

$$\begin{array}{lrcl} min & y = f(x) = [f_1(x), f_2(x), \cdots, f_m(x)] & & \\ s.t. & g_i(x) & \leq & 0, \quad i = 1, 2, \cdots, p \\ & h_j(x) & = & 0, \quad j = 1, 2, \cdots, q \end{array} \tag{1}$$

where $y = (f_1, f_2, \cdots, f_n) \in Y$ is the objective function vector, Y is the objective variable space, $g_i(x)$ is the i-th inequality constraint, and $h_j(x)$ is the j-th equality constraint.

Multiple objectives are included in MOPs, therefore, it is not possible to find a single solution that can optimize all objectives. Generally, improving one objective may cause the performance of the other objectives to decrease. Therefore, the conventional concept of single-objective optimality does not hold, and we must find a solution that is a compromize based on all objectives, i.e., the Pareto optimality. Based on the aforementioned reasons, some important definitions are given as follows for MOPs [18,19]:

Definition 1. *(Pareto dominance) The vector* $x' = (x'_1, x'_2, \cdots, x'_n)$ *dominates the vector* $x = (x_1, x_2, \cdots, x_n)$ *if and only if the next statement is verified.* $\forall i = (1, 2, \cdots, n) : f_i(x') \leq f_i(x)$ *and* $\exists i \in (1, 2, \cdots, n) : f_i(x') < f_i(x)$, *denoted as* $x' \prec x$.

Definition 2. *(Pareto optimality) A solution* $x^* \in D$ *is a Pareto optimal solution if there is not another* $x \in D$ *that satisfies* $f(x) \prec f(x^*)$.

Definition 3. *(Pareto optimal set) The Pareto optimal set is defined as the set of all Pareto optimal solutions.*

Definition 4. *(Pareto optimal front) The Pareto front consists of the values of the objectives corresponding to the solutions in the Pareto optimal set.*

3. An Introduction to the Multi-Objective Improved PSO

3.1. Main Aspects of the Standard PSO Algorithm

PSO was first presented by Kennedy and Eberhart in 1995 [20]. It is a random optimization algorithm based on swarm aptitude. The theory behind PSO comes from research on the behaviour of a bird swarm catching food. Compared with genetic algorithms, it has a simple construction, can be easily implemented, and has few adjustable parameters (Algorithm 1).

Let n be the dimension of the search space, $x_i = (x_{i1}, x_{i2}, \cdots, x_{in})$ be the current position of the i-th particle in the swarm, $p_{i\text{best}} = (p_{i\text{best}1}, p_{i\text{best}2}, \cdots, p_{i\text{best}n})$ be the best position of the i-th particle at that time, and $g_{\text{best}} = (g_{\text{best}1}, g_{\text{best}2}, \cdots, g_{\text{best}n})$ be the best position that the whole swarm has visited. The rate of the velocity of the i-th particle is denoted as $v_i = (v_{i1}, v_{i2}, \cdots, v_{in})$.

Algorithm 1: Standard particle swarm optimization [20]

Step 1: Initialize a population of particles X_N, such that each particle has a random position vector x_i and a velocity vector v_i. Set parameters c_1 and c_2, the maximum number of generations T_{max}, and the generation number $T = 0$.

Step 2: Calculate the fitness of all the particles in $X_N(T)$.

Step 3: Renew the positions and velocities of particles based on the following equations:

$$v_{id}^{T+1} = wv_{id}^{T} + c_1 r_1 (p_{i\text{best}d} - x_{id}^{T}) + c_2 r_2 (g_{\text{best}d} - x_{id}^{T}) \tag{2}$$

$$x_{id}^{T+1} = x_{id}^{T} + v_{id}^{T+1}. \tag{3}$$

Step 4: Calculate the fitness of the particles and renew every optimal position and global optimal position of the particles.

Step 5: (Termination examination) If the termination criterion is satisfied, then output the global optimal position and the fitness value. Otherwise, let $T = T + 1$ and return to **Step 2**.

3.2. Main Aspects of the Multi-Objective Improved PSO Algorithm (MOIPSO)

3.2.1. Elitist Archive and Crowding Entropy

Since Zitzler introduced SPEA with an elitist reservation mechanism in 1999 [5], many new algorithms have adopted a similar elitist reservation mechanism. Namely, they provide an external archive to store all the non-dominated solutions that have been found. The elitist reservation mechanism is also adopted in this article. As the evolution progresses, the non-dominated solutions in the current archive may not be the non-dominated solutions in the entire evolutionary process, therefore, the archive must be updated. The easiest updating method compares each solution with the current archive at each generation, which allows for the input of better solutions into the archive. The specific archive update rules are as follows [21]: (I) If the new solution dominates one or more solutions of the external archive, the new solution enters the archive, and the dominated solutions are deleted from the archive; (II) If the new solution is dominated by one or more solutions in the external archive, then the new solution is rejected; (III) If the new solution and the solutions in the external archive are not dominated by each other, then the new solution is a non-dominated solution and enters the archive. However, because of the storage space and computational efficiency, the external archive is not infinite. When the archive reaches its maximum size, the largest crowding degree solution will be deleted.

For the crowding distance measure, we cite the crowding entropy in the literature [21]. This method combines the crowded distance and distribution entropy, and the method accurately measures the crowding degree of the solution.

Crowding entropy is defined as follows:

$$\begin{aligned} CE_i &= \sum_{j=1}^{m} (c_{ij} E_{ij}) / (f_j^{\max} - f_j^{\min}) \\ &= -\sum_{j=1}^{m} [dl_{ij} \log_2(pl_{ij}) + du_{ij} \log_2(pu_{ij})] / (f_j^{\max} - f_j^{\min}), \end{aligned} \tag{4}$$

where E_{ij} is the distribution entropy of the i-th solution to the j-th objective function. Specifically, E_{ij} is defined as $E_{ij} = -[pl_{ij} \log_2(pl_{ij}) + pu_{ij} \log_2(pu_{ij})]$, where $pl_{ij} = \frac{dl_{ij}}{c_{ij}}$, $pu_{ij} = \frac{du_{ij}}{c_{ij}}$, and $c_{ij} = dl_{ij} + du_{ij}$. Variables dl_{ij} and du_{ij} are the distances from the i-th solution to the lower and upper adjacent solutions for the j-th objective function, $f_j^{\max}$ and $f_j^{\min}$ are the maximum and minimum values of the j-th objective function, and m is the number of objective functions.

Thus, the smaller the crowding entropy, the more crowded the archive. For each objective function, the boundary solutions are assigned infinite crowding entropy values. All other intermediate solutions are assigned crowding entropy values according to Equation (4).

3.2.2. Gaussian Mutation Strategy

Gaussian mutation is a very popular way to improve the particle swarm optimization algorithm. Higashi et al. [22] integrate a Gaussian mutation used for GA into PSO, and leave a certain ambiguity in the transition to the next generation due to Gaussian mutation. This method is used to solve the single-objective optimization problem, and carries on each individual variation in the current population. For the multi-objective problems, Coelho et al. [23] used Gaussian mutation to update the velocity update formula, but they only replaced the uniform random number R with a Gaussian random number Gd in the velocity formula. Liang et al. [24] also introduced Gaussian mutation, which will have a certain probability to initialize the particle adjacent to the target particle. Meanwhile, this will randomly initialize the particles beyond the range to increase the utilization rate of particles. To further improve the performance of the solutions from the MOPs, this paper presents a new Gaussian

mutation throw point strategy, which involves the throwing of points into external archives and the current population. The details of the strategy are as follows.

1. Throw points at sparse positions in the external archive to produce a thickened point set (TPS).

For MOPs, researchers hope that the non-dominated solution set is evenly distributed in the true Pareto front, but the solution sets of many methods yield uneven distributions. To increase the number of solutions at sparse positions and make the distribution of the solution set more uniform, we define the crowding degree of the solution in the external archive based on the crowding entropy and use throw points based on a Gaussian distribution of the largest crowding entropy solution (except the boundary solutions). Note that it is more important to find the boundary solutions for the MOPs, and the crowding entropy is infinite at the boundaries. Therefore, we must throw points at the boundary solutions.

The concrete operations are as follows.

Step 1: Identify a sparse solution based on the crowding entropy, as shown in Figure 1.

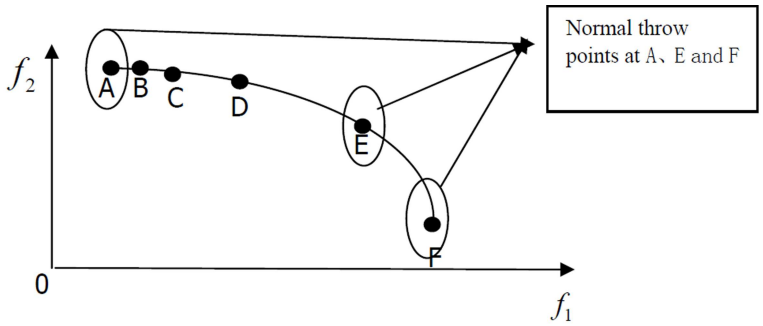

Figure 1. The selection process of the sparse solution.

Step 2: In the decision space, $A \longrightarrow x_A$, $E \longrightarrow x_E$, and $F \longrightarrow x_F$. We normally throw h points based on centres x_A, x_E, and x_F and variance σ. The random variable $Z \sim N(A, \sigma)$, and we add h random points to the TPS. It should be noted that the value of the variance σ is $\frac{1}{5}$ the width of each dimension. For example; $x = (x_1, x_2) \in [-10, 0] \times [0, 10] \Longrightarrow \sigma = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}; h = 5.$

2. Throw points into the current population to produce a Gaussian mutation points set (GMPS).

The algorithm chooses R solutions based on the prescribed probability in the current population, and normal throw points are established at R solutions (method based on (i): **Step 2**). Finally, we obtain a new GMPS.

3.2.3. Improved Learning Strategy

The standard PSO algorithm is used to solve the single-objective optimization problem, therefore, it is difficult to select the g_{best} value of velocity and update the formula for MOPs. The reason for this issue is that MOPs do not contain the global optimal solution. In many previous articles, researchers have randomly selected a solution from the external archive as the global optimal solution g_{best}, but this method lacks pertinence and cannot reflect the guidance of g_{best}. Therefore, this article presents a modified velocity formula, redefines the value g_{best} of Equation (2) and more efficiently applies g_{best} to solve MOPs.

(i) When x_i^T is not in the external archive, all solutions in the archive that dominate x_i^T can be regarded as global optimal solutions. Therefore, we provide a linear combination of these solutions.

Suppose that there are k archive solutions that dominate $x_i^T : a_1^T, a_2^T, \cdots, a_k^T$. A set of weights w_j is randomly generated, where $j = 1, 2, \cdots, k$ and $\sum_{j=1}^{k} w_j = 1$. Thus, g_{best} can be expressed as follows:

$$g_{\text{best}} = \sum_{j=1}^{k} w_j a_j^T.$$

Velocity is updated in the formula as follows:

$$v_{id}^{T+1} = w v_{id}^T + c_1 r_1 (p_{i\text{best}d} - x_{id}^T) + c_2 r_2 (g_{\text{best}d} - x_{id}^T). \tag{5}$$

(ii) When x_i^T is in the external archive, the concept of a global optimal solution is meaningless for x_i^T because the global optimal solution is a non-dominated solution. Therefore, all other solutions cannot be better than this solution, and g_{best} does not exist. Hence, the three parts of Equation (2) are unnecessary, and the velocity updating formula is as follows:

$$v_{id}^{T+1} = w v_{id}^T + c_1 r_1 (p_{i\text{best}d} - x_{id}^T). \tag{6}$$

The position update formula still uses the original model: $x_{id}^{T+1} = x_{id}^T + v_{id}^{T+1}$.

3.2.4. Update External Archive

The updating process of the external archive is an important problem for MOPs. Researchers typically use the archive update rules to compare the current population and the old external archive and then generate a new external archive to further improve the performance of the external archive. This paper uses three sets to update the old external archive. The specific updating methods are shown in Figure 2.

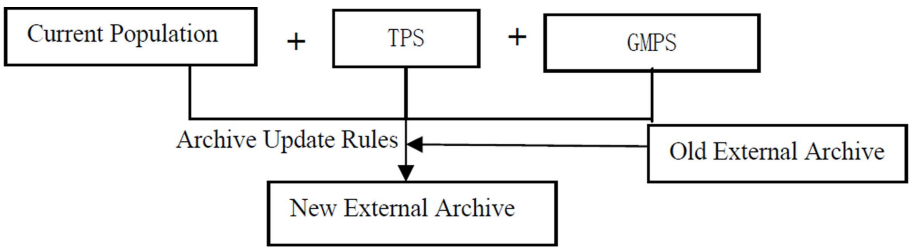

Figure 2. The updating process of the external archive.

3.2.5. Population Elitist Incremental Strategy

To increase the convergence rate of the population when the algorithm generates the offspring population, we consider the effect of not only the parent population but also the external archive. This paper proposes an elitist incremental strategy that increases the number of external archive solutions in the offspring population to form a new offspring population. The definition is as follows:

$$\begin{aligned} &\text{A new offspring population} \\ =&\text{randomly selected}(N - L)\text{offspring solutions} \\ &+ \text{randomly selected } L \text{ external archive solutions,} \end{aligned} \tag{7}$$

where N is the population size,

$$L = \begin{cases} T & T \leq \left[\frac{N}{2}\right] \,\&|A| \geq T \\ |A| & T \leq \left[\frac{N}{2}\right] \,\&|A| < T \\ \left[\frac{N}{2}\right] & T > \left[\frac{N}{2}\right] \,\&|A| \geq \left[\frac{N}{2}\right] \\ |A| & T > \left[\frac{N}{2}\right] \,\&|A| < \left[\frac{N}{2}\right] \end{cases},$$

T is the number of iterations, and $|A|$ is the external archive of size A at iteration T.

As a result, the population has strong exploration abilities and can find a wider range of non-dominated solutions at the early stage. The population will further explore the known non-dominated solutions and move closer to the true Pareto front in the later stage.

4. Overview of the MOIPSO Algorithm

As previously discussed, the MOIPSO algorithm can be summarized as follows.

Step l: Randomly initialize the position and velocity of each particle within the search space.

(a) Set the following parameters; $c_1 = c_2$, $v_{\max}$, $v_{\min}$, $w_{\max}$, $w_{\min}$, $T = 0$, the Gaussian mutation probability p_r, the maximum generation number $T_{\max}$, the population size N, and the maximum external archive size $A_{\max}$.

(b) Randomly initialize the population and the velocity.

Step 2: Calculate the fitness of the particles in the initialized population, and initialize the optimal position $p_{i\text{best}}$ of the i-th particle.

Step 3: Initialize an update to the external archive A.

Step 4: For $T = 1$:

(a) Renew the velocities of particles based on Equations (5) and (6) and the position of particles based on Equation (7) to form the middle population;

(b) Select the particles based on the Gaussian mutation probability p_r and throw points using the Gaussian mutation strategy (ii) to produce a GMPS;

(c) Calculate the crowding entropy of the external archive solutions and throw points based on the Gaussian mutation strategy (i) to produce a TPS;

(d) Renew the external archive A based on Section 3.2.4. If $|A| > A_{\max}$, then delete the most crowded particles according to the crowding entropy;

(f) Renew the middle population using the elitist incremental strategy, and form the new offspring population;

(g) Calculate the fitness of the particle in the offspring population;

(h) Renew the optimal position $p_{i\text{best}}$ of each particle;

(i) If the termination criterion is satisfied, then output the Pareto optimal solutions. Otherwise, let $T = T + 1$ and go to Step (a).

5. Methods and Simulation Experiments

5.1. Test Problems

To test the performance of MOIPSO, eight unconstrained optimization problems were used in the experiments. The SCH, KUR, and FON functions were suggested by Schaffer in 1985 [25], Kursawe in 1991 [26], and Fonseca in 1998 [27], respectively. The remainders are ZDT problems suggested by Zitzler et al. in 2000 [28]. The optimization problems are described in Table 1.

Table 1. The tested optimization problems.

Function	Objective Functions	D	Variable Bounds	Characteristics of the Pareto Front
SCH	$\begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$	1	$x \in [-10^3, 10^3]$	Convex
FON	$\begin{cases} f_1(x) = 1 - \exp(-\sum_{i=1}^{3} (x_i - \frac{1}{\sqrt{3}})^2) \\ f_2(x) = 1 - \exp(-\sum_{i=1}^{3} (x_i + \frac{1}{\sqrt{3}})^2) \end{cases}$	3	$x_i \in [-4, 4]$	Nonconvex
KUR	$\begin{cases} f_1(x) = \sum_{i=1}^{n-1} (-10exp(-0.2\sqrt{{x_i}^2 + {x_{i+1}}^2})) \\ f_2(x) = \sum_{i=1}^{n} (\lvert x_i \rvert^{0.8} + 5\sin({x_i}^3)) \end{cases}$	3	$x_i \in [-5, 5]$	Disconnect
ZDT1	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}] \\ g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1) \end{cases}$	30	$x_i \in [0, 1]$	Convex
ZDT2	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)[1 - (x_1/g(x))^2] \\ g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1) \end{cases}$	30	$x_i \in [0, 1]$	Nonconvex
ZDT3	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)(1 - \sqrt{x_1/g(x)} - x_1 \sin(10\pi x_1)/g(x)) \\ g(x) = 1 + 9\sum_{i=2}^{n} x_i/(n-1) \end{cases}$	30	$x_i \in [0, 1]$	Convex disconnect
ZDT4	$\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)(1 - \sqrt{x_1/g(x)}) \\ g(x) = 1 + 10(n-1) + \sum_{i=2}^{n} [{x_i}^2 - 10\cos(4\pi x_i)] \end{cases}$	10	$x_1 \in [0, 1]$ $x_i \in [-5, 5]$ $i = 2, \cdots, n$	Nonconvex
ZDT6	$\begin{cases} f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(x) = g(x)[1 - (f_1(x)/g(x))^2] \\ g(x) = 1 + 9[\sum_{i=2}^{n} x_i/(n-1)]^{0.25} \end{cases}$	10	$x_i \in [0, 1]$	Nonconvex

5.2. Performance Measures

The standard performance measures of multi-objective evolutionary algorithms were used to evaluate the performance of the proposed algorithm. The performance measures are briefly described as follows.

5.2.1. Convergence Measure Indicator

Ideally, the iterative process of MOEA approaches the Pareto front, but in most cases, it is difficult to find the true Pareto front. The proximity of the approximate solutions to the Pareto optimal solutions is a main indicator.

The concept of generational distance was introduced by Van Veldhuizen [29] to measure the proximity of the approximate solutions to the Pareto optimal solutions. This indicator is defined as follows:

$$GD = \frac{\sqrt{\sum_{i=1}^{n} \text{dist}_i^2}}{n}, \tag{8}$$

where n is the number of non-dominated solutions. When the Pareto fronts of the objective function can be expressed in analytic form, dist_i is measured by the Euclidean distance (in objective space) between the i-th non-dominated solution and the nearest member of the Pareto optimal set. Otherwise, dist_i is measured by the Euclidean distance between the i-th non-dominated solution and the reference

point set. It is clear that a smaller value of GD is better, and $GD = 0$ indicates that the non-dominated solution set is located in the true Pareto front.

5.2.2. Distribution Measure Indicator

Typically, we hope that non-dominated solutions are uniformly distributed in the true Pareto front. Two main factors used to measure the distribution are uniformity and width:

(1) Distribution uniformity (Δ)

The indicator Δ [4] is used to measure the uniformity and diversity of the non-dominated solution set. When calculating this indicator, we must sort the obtained non-dominated solutions based on the specified objective function values. This indicator is defined as follows:

$$\Delta = \frac{h_f + h_l + \sum_{i=1}^{n-1} |h_i - \bar{h}|}{h_f + h_l + (n-1)\bar{h}}, \tag{9}$$

where n is the number of non-dominated solutions, h_i is the Euclidean distance between neighbouring solutions in the non-dominated solution set, $\bar{h}$ is the mean of all h_i values, h_f and h_l are the Euclidean distances between the extreme solutions of the Pareto optimal solution set and the boundary solutions of the non-dominated solution set. When the non-dominated solution set is uniformly distributed in the true Pareto front, $h_f = 0$, $h_l = 0$, and all $h_i = \bar{h}$, therefore, $\Delta = 0$. A small value of Δ indicates better uniformity in the true Pareto front.

(2) Distribution width (*DW*)

Generally, it is favourable if the boundary solutions can be included in the non-dominated solution set. In other words, in these types of problems, researchers hope to find the boundary points of the true Pareto fronts.

The indicator $M_3^*(NP)$, which measures the distribution width, was proposed by Zitzler et al. [28]. The associated formula is as follows:

$$M_3^*(NP) = \sqrt{\sum_{i=1}^{M} \max\{\| p_i - q_i \|^*, p, q \in NP\}}, \tag{10}$$

where NP is the non-dominated solution set and M is the dimension of the non-dominated solutions. Notably, M_3^* can measure the distribution width of the non-dominated solution set, but when the distribution range of the Pareto front is too large or the dimensions of the solutions are large, the value of M_3^* will be large. Therefore, it is difficult to compare and compile the data.

Based on the aforementioned method, the new indicator, DW, is provided. The concrete form of this indicator is as follows:

$$DW = \left| \frac{\prod_{i=1}^{M} \max\{|p_i - q_i|, p, q \in NP\}}{\prod_{i=1}^{M} \max\{|p'_i - q'_i|, p', q' \in RP\}} - 1 \right|, \tag{11}$$

where RP is the Pareto optimal set or the reference point set. Notably, a small value of DW reflects a better distribution uniformity for the non-dominated solution set.

5.3. Algorithm Comparison

To validate the MOIPSO algorithm, we compared it to NSGA-II [4] and MOPSO [7] based on the above three performance measures. The source codes of NSGA-II and MOPSO are available at http://delta.cs.cinvestav.mx/~ccoello/EMOO/ (matlab code).

The initial population size is 100 for MOIPSO, NSGA-II and MOPSO. The number of iterations directly affects the time complexity of the algorithm, and the convergence of the problem with different iterations is discussed [30]. It can be found that when the program reaches a stable state, the subsequent iterations can no longer improve the performance of the algorithm, but only increase the running time. Therefore, through a large number of numerical experiments, different iteration numbers were chosen based on the complexity of the problems in the three algorithms. The number of iterations completed was 60 for SCH, 100 for FON and KUR, 300 for ZDT1, ZDT2, and ZDT3, and 1000 for ZDT4 and ZDT6.

The range of the parameter has been discussed in some literature [31], so we took the commonly used parameter values for NSGAII and MOPSO. To make the test and its results more comparable, for MOIPSO, the same parameters as MOPSO took the same values, and other parameters were set after a large number of numerical experiments. In the MOPSO and MOIPSO algorithms, $c_1 = c_2 = 1.7$, while r_1 and r_1 are assigned random values between 0 and 1. In MOPSO, the inertia weight is $w = 0.7$, and the inertia weight damping ratio $w_{damp} = 1$. In MOIPSO, the inertia weight w adaptively decreased from $w_{\max} = 0.9$ to $w_{\min} = 0.4$ according to the following formula: $w = w_{\max} - \frac{T(w_{\max} - w_{\min})}{T_{\max}}$. In the NSGA-II [4] algorithm, the crossover probability is 0.8, and the mutation probability is 0.3. To evaluate the statistical performance, all the experiments are run 30 times. The best, worst, mean, and average deviations are shown in Tables 2–4, respectively.

The *GD* results are shown in Table 2. MOIPSO exhibits the best *GD* values for SCH, KUR, ZDT1 and ZDT4. MOPSO displays the best *GD* values for ZDT2, ZDT3, and ZDT6. FON, MOIPSO, and MOPSO exhibit similar results. For the worst *GD*, MOIPSO displays high stability and consistently yields the lowest worst *GD* value in all test problems. For the mean *GD*, MOIPSO produces the best mean values for all test problems except ZDT4, for which FON, MOIPSO, and MOPSO exhibit similar results. With respect to the standard deviation of *GD*, MOIPSO exhibits the best solution for KUR, ZDT1, ZDT2, and ZDT3. NSGA-II yields the best results for the other functions. Thus, MOIPSO produces better values of *GD* indicators than the other two algorithms in most test problems, and the results of MOIPSO are better than those of the other algorithms by 1∼2 orders of magnitude. This finding indicates that the resulting Pareto fronts obtained via MOIPSO are closer to the true Pareto fronts, and MOIPSO can effectively improve convergence.

Some information for Δ is shown in Table 3. For the SCH and ZDT1 functions, all the solutions of MOIPSO are better than those of the other algorithms. MOPSO exhibits the best, and the mean best, Δ value for KUR. MOIPSO has the minimal worst Δ and the best standard deviation. MOIPSO has the best, the mean best, and the minimal worst Δ values for ZDT2 and ZDT3, but the standard deviation of NSGA-II is the best. MOIPSO displays the best and the mean best Δ values for ZDT4. NSGA-II exhibits the minimal worst Δ, and MOPSO yields the best standard deviation. MOPSO provides the best Δ for ZDT6. MOIPSO has the minimal worst Δ and the mean best Δ, and the minimal standard deviation of NSGA-II is the best. Table 3 shows that MOIPSO provides the best mean solution for all seven functions. Therefore, the MOIPSO results are evenly distributed in the experiments, however, they are not shown for all the functions.

Table 4 shows the results of a new quality indicator—*DW*. All the solutions of MOIPSO are better than those of the other algorithms for SCH, KUR, and ZDT1. MOIPSO yields the three best indicators for the FON function, and MOPSO provides the best solution for *DW*. NSGA-II exhibits the best *DW* solution for ZDT2, and the other indicators of MOIPSO are the best. MOIPSO displays the best and the mean best *DW* for ZDT3, and NSGA-II yields the minimal worst *DW* and the best standard deviation. NSGA-II exhibits the best *DW* for ZDT4, and MOPSO displays the minimal worst *DW*. MOIPSO produces the mean best *DW* and the minimal standard deviation. For the ZDT6 function, NSGA-II exhibits the minimal standard deviation, and the other indicators of MOIPSO are the best. Similarly, the *DW* results in Table 4 show that MOIPSO is able to produce the best distribution of solutions in the Pareto optimal front for most of the test functions.

Table 2. Comparison of the results of multi-objective improved particle swarm optimization algorithm (MOIPSO) and different algorithms based on the indicator, GD.

Function	Statistic	MOPSO	NSGA-II	MOIPSO
SCH	Best	8.72×10^{-4}	8.37×10^{-4}	8.34×10^{-4}
	Worst	1.30×10^{-3}	1.10×10^{-3}	1.00×10^{-3}
	Mean	9.66×10^{-4}	9.56×10^{-4}	9.41×10^{-4}
	Std	7.89×10^{-5}	4.70×10^{-5}	4.76×10^{-5}
FON	Best	1.00×10^{-3}	1.10×10^{-3}	1.00×10^{-3}
	Worst	1.30×10^{-3}	1.30×10^{-3}	1.20×10^{-3}
	Mean	1.10×10^{-3}	1.20×10^{-3}	1.10×10^{-3}
	Std	6.22×10^{-5}	3.73×10^{-5}	6.08×10^{-5}
KUR	Best	3.10×10^{-3}	1.60×10^{-3}	1.40×10^{-3}
	Worst	3.09×10^{-2}	4.40×10^{-3}	3.80×10^{-3}
	Mean	5.80×10^{-3}	3.00×10^{-3}	2.50×10^{-3}
	Std	4.82×10^{-3}	6.43×10^{-4}	4.13×10^{-4}
ZDT1	Best	1.70×10^{-3}	1.96×10^{-2}	5.91×10^{-4}
	Worst	6.43×10^{-2}	5.49×10^{-2}	3.20×10^{-3}
	Mean	2.63×10^{-2}	3.03×10^{-2}	1.00×10^{-3}
	Std	1.81×10^{-2}	7.80×10^{-3}	6.30×10^{-4}
ZDT2	Best	5.28×10^{-5}	2.96×10^{-2}	3.51×10^{-4}
	Worst	1.55×10^{-1}	4.99×10^{-2}	3.10×10^{-3}
	Mean	4.19×10^{-2}	3.84×10^{-2}	7.52×10^{-4}
	Std	4.49×10^{-2}	5.40×10^{-3}	5.94×10^{-4}
ZDT3	Best	4.04×10^{-4}	2.19×10^{-2}	5.01×10^{-4}
	Worst	1.08×10^{-1}	4.74×10^{-2}	7.10×10^{-3}
	Mean	3.82×10^{-2}	3.30×10^{-2}	1.20×10^{-3}
	Std	3.27×10^{-2}	8.10×10^{-3}	1.50×10^{-3}
ZDT4	Best	1.10×10^{-1}	1.30×10^{-3}	3.23×10^{-4}
	Worst	2.01×10^{-1}	2.86×10^{-2}	1.61×10^{-2}
	Mean	1.40×10^{-1}	4.90×10^{-3}	9.90×10^{-3}
	Std	1.67×10^{-2}	6.20×10^{-3}	2.37×10^{-2}
ZDT6	Best	2.72×10^{-4}	1.19×10^{-1}	5.36×10^{-4}
	Worst	3.32×10^{-1}	1.65×10^{-1}	4.38×10^{-2}
	Mean	2.29×10^{-2}	1.46×10^{-1}	1.09×10^{-2}
	Std	5.96×10^{-2}	9.10×10^{-3}	1.13×10^{-2}

Table 3. Comparison of the results of MOIPSO and different algorithms based on Δ.

Function	Statistic	MOPSO	NSGA-II	MOIPSO
SCH	Best	4.68×10^{-1}	4.68×10^{-1}	4.67×10^{-1}
	Worst	4.80×10^{-1}	4.73×10^{-1}	4.70×10^{-1}
	Mean	4.68×10^{-1}	4.69×10^{-1}	4.67×10^{-1}
	Std	2.80×10^{-3}	1.70×10^{-3}	4.53×10^{-4}
FON	Best	4.82×10^{-1}	4.81×10^{-1}	4.84×10^{-1}
	Worst	4.94×10^{-1}	4.95×10^{-1}	4.85×10^{-1}
	Mean	4.86×10^{-1}	4.87×10^{-1}	4.85×10^{-1}
	Std	2.60×10^{-3}	3.90×10^{-3}	4.70×10^{-4}
KUR	Best	4.59×10^{-1}	4.64×10^{-1}	4.63×10^{-1}
	Worst	4.69×10^{-1}	4.69×10^{-1}	4.68×10^{-1}
	Mean	4.63×10^{-1}	4.67×10^{-1}	4.66×10^{-1}
	Std	2.10×10^{-3}	1.30×10^{-3}	1.10×10^{-3}
ZDT1	Best	4.08×10^{-2}	2.67×10^{-1}	1.13×10^{-2}
	Worst	3.54×10^{-1}	4.05×10^{-1}	1.08×10^{-1}
	Mean	2.47×10^{-1}	3.43×10^{-1}	4.28×10^{-2}
	Std	8.62×10^{-2}	3.71×10^{-2}	9.70×10^{-3}

Table 3. *Cont.*

Function	Statistic	MOPSO	NSGA-II	MOIPSO
ZDT2	Best	3.78×10^{-1}	4.09×10^{-1}	3.90×10^{-3}
	Worst	9.92×10^{-1}	4.90×10^{-1}	1.47×10^{-1}
	Mean	6.88×10^{-1}	4.59×10^{-1}	2.65×10^{-2}
	Std	1.54×10^{-1}	1.99×10^{-2}	3.17×10^{-2}
ZDT3	Best	4.19×10^{-1}	2.64×10^{-1}	2.50×10^{-3}
	Worst	7.06×10^{-1}	3.60×10^{-1}	1.94×10^{-1}
	Mean	5.58×10^{-1}	3.13×10^{-1}	3.08×10^{-2}
	Std	6.58×10^{-2}	2.94×10^{-2}	4.57×10^{-2}
ZDT4	Best	6.11×10^{-1}	1.33×10^{-2}	3.70×10^{-3}
	Worst	6.89×10^{-1}	3.50×10^{-1}	4.74×10^{-1}
	Mean	6.50×10^{-1}	8.96×10^{-2}	8.88×10^{-2}
	Std	1.95×10^{-2}	9.90×10^{-2}	1.50×10^{-1}
ZDT6	Best	1.07×10^{-2}	4.86×10^{-1}	1.11×10^{-2}
	Worst	4.52×10^{-1}	7.95×10^{-1}	4.08×10^{-1}
	Mean	2.43×10^{-1}	6.29×10^{-1}	2.28×10^{-1}
	Std	1.28×10^{-1}	6.92×10^{-2}	1.28×10^{-1}

Table 4. Comparison of the results of MOIPSO and different algorithms based on *DW*.

Function	Statistic	MOPSO	NSGA-II	MOIPSO
SCH	Best	1.00×10^{-3}	4.46×10^{-5}	4.44×10^{-5}
	Worst	6.43×10^{-2}	4.35×10^{-2}	1.80×10^{-2}
	Mean	1.43×10^{-2}	1.66×10^{-2}	2.70×10^{-3}
	Std	1.56×10^{-2}	1.26×10^{-2}	3.70×10^{-3}
FON	Best	2.62×10^{-4}	4.21×10^{-4}	1.90×10^{-3}
	Worst	3.36×10^{-2}	9.09×10^{-2}	8.00×10^{-3}
	Mean	9.60×10^{-3}	3.10×10^{-2}	5.50×10^{-3}
	Std	9.30×10^{-3}	2.19×10^{-2}	1.60×10^{-3}
KUR	Best	3.07×10^{-4}	5.50×10^{-3}	2.40×10^{-4}
	Worst	7.55×10^{-2}	7.36×10^{-2}	1.11×10^{-2}
	Mean	1.32×10^{-2}	3.71×10^{-2}	3.30×10^{-3}
	Std	1.82×10^{-2}	1.95×10^{-2}	2.40×10^{-3}
ZDT1	Best	7.80×10^{-3}	5.00×10^{-2}	3.20×10^{-3}
	Worst	1.33	1.29	2.26×10^{-2}
	Mean	5.23×10^{-1}	4.98×10^{-1}	1.17×10^{-2}
	Std	4.05×10^{-1}	2.83×10^{-1}	6.20×10^{-2}
ZDT2	Best	1.54×10^{-1}	1.60×10^{-3}	3.30×10^{-3}
	Worst	1.00	6.40×10^{-1}	2.99×10^{-1}
	Mean	7.72×10^{-1}	2.07×10^{-1}	4.16×10^{-2}
	Std	2.55×10^{-1}	1.49×10^{-1}	6.28×10^{-2}
ZDT3	Best	4.60×10^{-3}	3.49×10^{-2}	1.83×10^{-4}
	Worst	8.93×10^{-1}	5.54×10^{-1}	9.63×10^{-1}
	Mean	6.23×10^{-1}	2.59×10^{-1}	8.52×10^{-2}
	Std	2.37×10^{-1}	1.54×10^{-1}	2.37×10^{-1}
ZDT4	Best	1.92×10^{-1}	9.28×10^{-4}	6.50×10^{-3}
	Worst	1.00	1.64	1.68
	Mean	6.33×10^{-1}	2.21×10^{-1}	2.07×10^{-1}
	Std	1.95×10^{-1}	4.03×10^{-1}	1.55×10^{-1}
ZDT6	Best	1.42×10^{-2}	1.04×10^{-2}	2.20×10^{-4}
	Worst	6.87	1.17	1.09
	Mean	1.18	2.78×10^{-1}	1.86×10^{-1}
	Std	1.48	2.78×10^{-1}	8.77×10^{-1}

According to the above statistical analyses, MOIPSO successfully solves the SCH, ZDT1 and ZDT6 problems, as illustrated by Figures 3–5. Figure 3 shows that the three algorithms have

similar convergence performances for the SCH function, however, MOIPSO exhibits better uniformity. Figures 4 and 5 illustrate the superior convergence and distribution results of MOIPSO compared with those of the other algorithms. Furthermore, by combining the data from Table 4 and the figures, we can see that when the boundary solutions do not exist in non-dominated solutions or solutions do not converge to the true Pareto front, then the *DW* value is very large. In the case of NSGAII and MOPSO for ZDT6, because some points are far away from the true optimal pare fronts, their *DW* values are 10^{-2}, but the values are 10^{-4} for MOIPSO and we can find that all the solutions are near the true optimal Pareto fronts in Figure 4. Thus, the *DW* indicator can measure the distribution width of the non-dominated solution set.

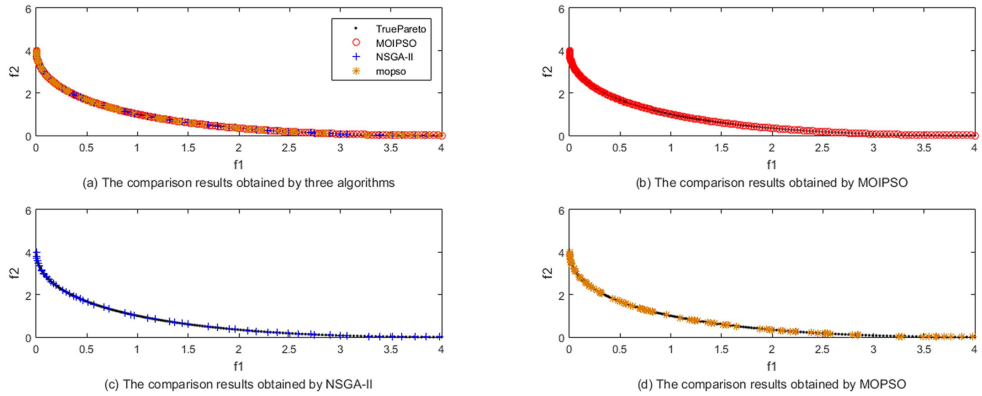

Figure 3. For SCH, the comparisons between the true Pareto front and the best ones obtained by three different algorithms.

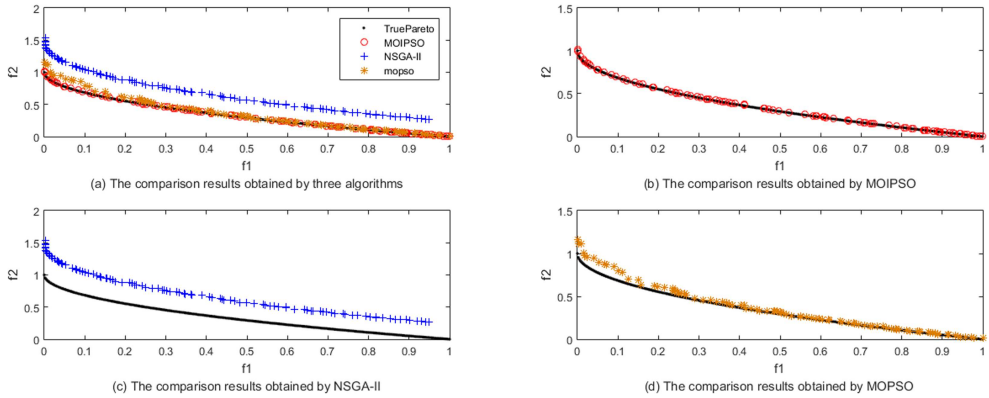

Figure 4. For ZDT1, the comparisons between the true Pareto front and the best ones obtained by three different algorithms.

From the numerical results, we can also see that the MOIPSO algorithm performs better than other algorithms in distribution uniformity and width. This is obviously a good result of the Gaussian mutation throw point strategy. However, this method is not omnipotent. Firstly, when the number of throwing points is too large, the running time will rapidly increase. It is difficult to determine the number, so in this paper, we have done a lot of experiments to determine it. Secondly, for all sparse solutions, we will throw points, so when the Pareto optimal front of optimization problem is very complex and contains a large number of outliers, the effectiveness of our Gaussian mutation throw point strategy will be affected.

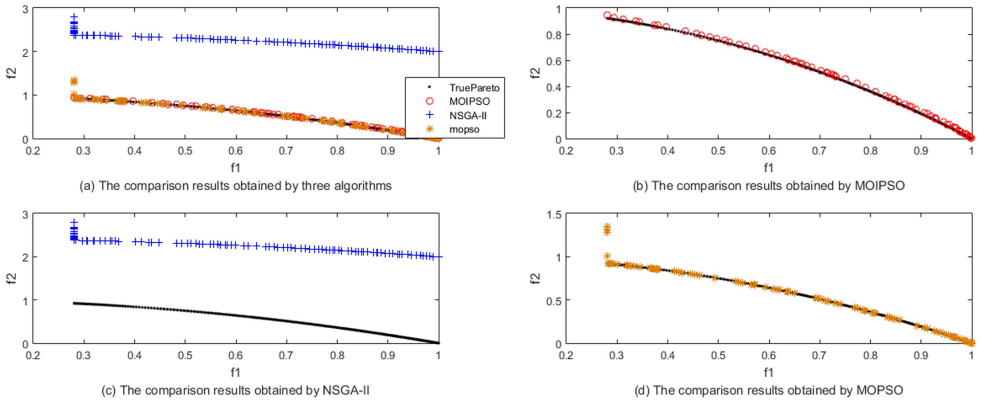

Figure 5. For ZDT6, the comparisons between the true Pareto front and the best ones obtained by three different algorithms.

6. Conclusions

A new multi-objective PSO algorithm based on Gaussian mutation and an improved learning strategy (MOIPSO) is presented to solve MOPs. First, MOIPSO builds different learning strategies to update the individual positions of the non-dominated and dominated solutions. Then, the Gaussian mutation strategy is used to create throw points at sparse and boundary positions. These updating strategies yield high convergence and a satisfactory distribution. To further measure the distribution width, the indicator *DW* is proposed.

The performance of MOIPSO was tested based on different MOP benchmark functions with convex and nonconvex objection functions. To demonstrate the effectiveness of MOIPSO, the results were compared to those of MOPSO and NSGA-II. The experimental results showed that MOIPSO significantly outperforms all other algorithms based on the test problems with respect to three metrics. The resulting data and figures indicate that the proposed *DW* indicator is reasonable.

In this article, only two-objective functions are tested. In the near future, we also plan to evaluate MOIPSO using other objective test functions. Furthermore, most parameters in this paper (such as the number of throw points, cognitive coefficient, etc.) have certain values. It would be interesting to study whether these control parameters could adaptively change as the iteration time increases.

Author Contributions: Both authors have contributed equally to this paper. Both authors have read and approved the final manuscript.

Funding: This research was funded by the National Natural Science Foundation of P.R. China (61561001) and First-Class Disciplines Foundation of NingXia (Grant No. NXYLXK2017B09).

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

Abbreviations

The following abbreviations are used in this manuscript:

MOIPSO	Multi-objective improved PSO algorithm
MOPs	Multi-objective optimization problems
MOEA	Multi-objective optimization evolutionary algorithm
GMPS	Gaussian mutation points set
TPS	Thickened point set
DW	Distribution width

References

1. Miettinen, K.M. *Nonlinear Multi-Objective Optimization*; Kluwer Academic Publishers: Boston, MA, USA, 1999.
2. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; John Wiley and Sons: Hoboken, NJ, USA, 2001.
3. Fonseca, C.M.; Fleming, P.J. *Genetic Algorithm for Multi-Objective Optimization: Formulation, Discussion and Generalization*; Morgan Kaufmann Publishers: Burlington, MA, USA, 1993.
4. Deb, K.; Pratap, A.; Agrawal, S.; Meyarivan, T. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
5. Zitzler, E.; Thiele, L. Multi-objective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
6. Knowles, J.D.; Corne, D.W. Approximating the nondominated front using the Pareto archived evolution strategy. *IEEE Trans. Evol. Comput.* **2000**, *8*, 149–172. [CrossRef]
7. Coello, C.A.C.; Pultdo, G.T.; Lechuga, M.S. Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 256–279. [CrossRef]
8. Sierra, M.R.; Coello, C.A.C. *Improving PSO-Based Multi-Objective Optimization Using Crowding, Mutation and ε-Dominance*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 505–519.
9. Reddy, M.J.; Kumar, D.N. An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design. *Eng. Optim.* **2007**, *39*, 49–68. [CrossRef]
10. Leong, W.F.; Yen, G.G. PSO-based multi-objective optimization with dynamic population size and adaptive local archives. *IEEE Trans. Syst. Man Cybern. Part B* **2008**, *38*, 1270–1293. [CrossRef] [PubMed]
11. Yen, G.G.; Leong, W.F. Dynamic multiple swarms in multi-objective particle swarm optimization. *IEEE Trans. Syst. Man Cybern. Part B* **2009**, 39, 890–911. [CrossRef]
12. Chen, D.B.; Zou, F.; Wang, J.T. A multi-objective endocrine PSO algorithm and application. *Appl. Soft Comput.* **2011**, *11*, 4508–4520. [CrossRef]
13. Lin, Q.; Li, J.; Du, Z.; Chen, J.; Ming, Z. A novel multi-objective particle swarm optimization with multiple search strategies. *Eur. J. Oper. Res.* **2015**, *247*, 732–744. [CrossRef]
14. Meza, J.; Espitia, H.; Montenegro, C.; Giménez, E.; González-Crespo, R. Movpso: Vortex multi-objective particle swarm optimization. *Appl. Soft Comput.* **2017**, *52*, 1042–1057. [CrossRef]
15. Hu, W.; Yen, G.G. Adaptive multi-objective particle swarm optimization based on parallel cell coordinate system. *IEEE Trans. Evol. Comput.* **2015**, *19*, 1–18.
16. Knight, J.T.; Singer, D.J.; Collette, M.D. Testing of a spreading mechanism to promote diversity in multi-objective particle swarm optimization. *IEEE Trans. Evol. Comput.* **2015**, *16*, 279–302. [CrossRef]
17. Cheng, T.; Chen, M.; Fleming, P.J.; Yang, Z.; Gan, S. A novel hybrid teaching learning based multi-objective particle swarm optimization. *Neurocomputing* **2017**, *222*, 11–25. [CrossRef]
18. Coello, C.A.C. Evolutionary multi-objective optimization: A historical view of the field. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–36. [CrossRef]
19. Zitzler, E. *Evolutionary Algorithm for Multiobjective Optimization: Methods and Application*; Swiss Federal Institute of Technology: Zurich, Switzerland, 1999.
20. Kennedy, J.; Eberhart, R.C. Particle Swarm Optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
21. Wang, Y.N.; Wu, L.H.; Yuan, X.F. Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure. *Soft Comput.* **2010**, *14*, 193–209. [CrossRef]
22. Higashi, N.; Iba, H. *Particle Swarm Optimization with Gaussian Mutation*; IEEE: Piscataway, NJ, USA, 2013.
23. Coelho, L.D.S.; Ayala, V.H.; Alotto, P. A Multiobjective Gaussian Particle Swarm Approach Applied to Electromagnetic Optimization. *IEEE Trans. Mag.* **2010**, *46*, 3289–3292. [CrossRef]
24. Liang, H.M.; Zhang, K.; Yu, H.H. Multi-objective Gaussian particle swarm algorithm optimization based on niche sorting for actuator design. *Adv. Mech. Eng.* **2015**, *7*, 1–7. [CrossRef]
25. Schaffer, J.D. Multiple objective optimization with vector evaluated genetic algorithm. In Proceedings of the 1st International Conference on Genetic Algorithm and Their Applications, Pittsburg, CA, USA, 24–26 July 1985; pp. 93–100.
26. Kursawe, F. A varant of evolution strategies for vector optimization. In *International Conference on Parallel Problem Solving from Nature*; Springer: Berlin/Heidelberg, Germany, 1991; pp. 193–197.

27. Fonseca, C.M.; Fleming, P.J. Multiobjective optimization and multiple constraint handling with evolutionary algorithms, Part II: Application example. *IEEE Trans. Syst. Man Cybern. A* **1998**, *28*, 38–47. [CrossRef]
28. Zitzler, E.; Deb, K.; Thiele, L. Comparison of multi-objective evolutionary algorithms: Empirical results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef]
29. Van Veldhuizen, D.A.; Lamont, G.B. Evolutionary computation and convergence to a Pareto front. In *Late Breaking Papers at the Genetic Programming 1998 Conference*; Standford University: Stanford, CA, USA, 1998; pp. 221–228.
30. Saraswat, A.; Saini, A. Multi-objective optimal reactive power dispatch considering voltage stability in power systems using HFMOEA. *Eng. Appl. Artif. Intell.* **2013**, *26*, 390–404. [CrossRef]
31. Ding, S.X.; Chen, C.; Xin, B.; Psrdalos, P. A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches. *Appl. Soft Comput.* **2018**, *63*, 249–267. [CrossRef]