mathematics

Article

A Parallel Compact Gannet Optimization Algorithm for Solving Engineering Optimization Problems

Jeng-Shyang Pan [1,2], Bing Sun [1], Shu-Chuan Chu [1,*], Minghui Zhu [1] and Chin-Shiuh Shieh [3]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2] Department of Information Management, Chaoyang University of Technology, Taichung 41349, Taiwan
[3] Department of Electronic Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 80778, Taiwan
* Correspondence: scchu0803@gmail.com

Abstract: The Gannet Optimization Algorithm (GOA) has good performance, but there is still room for improvement in memory consumption and convergence. In this paper, an improved Gannet Optimization Algorithm is proposed to solve five engineering optimization problems. The compact strategy enables the GOA to save a large amount of memory, and the parallel communication strategy allows the algorithm to avoid falling into local optimal solutions. We improve the GOA through the combination of parallel strategy and compact strategy, and we name the improved algorithm Parallel Compact Gannet Optimization Algorithm (PCGOA). The performance study of the PCGOA on the CEC2013 benchmark demonstrates the advantages of our new method in various aspects. Finally, the results of the PCGOA on solving five engineering optimization problems show that the improved algorithm can find the global optimal solution more accurately.

Keywords: Gannet Optimization Algorithm; parallel communication strategy; compact strategy; engineering optimization

MSC: 68T20

check for updates

Citation: Pan, J.-S.; Sun, B.; Chu, S.-C.; Zhu, M.; Shieh, C.-S. A Parallel Compact Gannet Optimization Algorithm for Solving Engineering Optimization Problems. *Mathematics* **2023**, *11*, 439. https://doi.org/10.3390/math11020439

Academic Editors: Yu Xue, Chunlin He, Ferrante Neri and Simeon Reich

Received: 30 November 2022
Revised: 6 January 2023
Accepted: 9 January 2023
Published: 13 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid development of modern industry, optimization theory and optimization methods have spread throughout all aspects of industrial production, and in recent years, the efficiency and accuracy of optimization solutions are more and more strict. Optimization problems exist in all walks of life around us, such as engineering optimization, neural network optimization, intelligent computing, and some path planning problems that require a highly efficient and accurate optimization algorithm to solve. It has been found that various optimization problems can be solved effectively by meta-heuristic algorithms [1], such as in the field of QR code technology [2]. Population intelligence algorithm is a kind of meta-heuristic algorithm, and population intelligence algorithm is an algorithm studied based on natural organisms or natural phenomenon, such as Particle Swarm Optimization Algorithm (PSO) [3–6], Cuckoo Search Algorithm (CSA) [7,8], Ant Colony Optimization Algorithm (ACO) [9–11], Flower Pollination Algorithm (FPA) [12–14], Phasmatodea Population Evolution Algorithm (PPE) [15], Symbiotic Organisms Search (SOS) [16,17], etc.

The Gannet Optimization Algorithm (GOA) is based on a summary of the patterns of fish predation of natural organisms gannets [18]. The Gannet Optimization Algorithm (GOA) is simple in structure and easy to understand. There are two stages of the GOA: the exploration stage of the fish when the Gannet hunt for food and the exploitation stage that unfolds when the Gannet find the fish and chase them. The GOA randomly selects these two modes to start feeding on the fish in each iteration, and this strategy allows global exploitation and local exploration to be performed randomly throughout the process.

Although the GOA has a better search capability than algorithms such as PSO, the large amount of memory it occupies in the process of finding the optimum is still a major drawback that limits the convergence efficiency of the GOA. There are many ways to improve the performance of the algorithm, such as adopting surrogate-assisted strategy [19,20], adaptive strategy [21–23] and so on. For algorithmic memory saving strategies, many compact schemes have been proposed in other algorithms, such as compact Genetic Algorithm (cGA) [24], compact Differential Evolution Algorithm (cDE) [25], compact Particle Swarm Optimization Algorithm (cPSO) [26], etc. However, the GOA does not currently have other options for the compact improvement strategy. The improved GOA with the compact strategy not only improves the convergence of the GOA but also saves the memory usage of the computer, which is called the cGOA. Although the cGOA has a fast convergence speed, the accuracy of its convergence has no advantage compared with the original algorithm. We try to improve the stability of the algorithm by adding a parallel communication strategy to the improved compact strategy, and call the improved algorithm the PCGOA.

Based on the above description, the improved GOA can both save computer memory and solve optimization problems more accurately. Finally, the PCGOA is applied to the selected five engineering optimization problems. This paper contributes as follows:

- For the shortcomings of memory occupation and convergence efficiency of the GOA, this paper proposes a GOA with a combined strategy of parallel and compact, and the improved algorithm is called the PCGOA.
- Two new parallel communication strategies are proposed in parallel strategies to improve the performance of the algorithm.
- In this paper the proposed parallel compact GOA uses the test function CEC2013 to compare with some traditional algorithms, such as PSO algorithm, SCA algorithm, PMVO algorithm, etc. It is proved that the PCGOA has better performance.
- The improved GOA algorithm was applied to five engineering optimization problems, and the results indicated that not only the convergence speed was improved, but also a large amount of computer memory was saved.

The sections of this paper are organized as follows. The related work in Section 2 briefly reviews the basic principles of the original GOA and the principles of Compact Scheme. Section 3 specifies the improvement process of the GOA and the principle of the PCGOA, and proposes two new communication strategies. Section 4 tests the performance of the improved algorithm based on the CEC2013 benchmark function and analyzes the data image curves. Section 5 explains the method of improving the algorithm to incorporate five engineering optimization problems and analyzes the performance of the improved algorithm. Finally, the entire article process is summarized in Section 5.

2. Related Works

There are two main parts in this section. The first part briefly reviews the basic principles of the GOA. The second part mainly introduces the compact scheme.

2.1. Gannet Optimization Algorithm

The GOA simulates a process of Gannet in a lake from finding a target to feeding on it. The Gannet is a large waterfowl that is ideally suited to feeding on targets in the water due to its size, and is able to grab targets and bring them out of the water at a faster rate during the feeding process. The Gannet also know how to work as a team. When flocks of gannets find schools of fish, they will line up or feed on them in a semicircle. The GOA is a simulation of the specific process of Gannet feeding on targets in the water.

The GOA has two stages of exploration and development. The first initialization of the GOA is to define a set of random solutions x_{id} representing n D-dimensional gannets locations of n*D matrices to start with, and the optimal solution obtained from this matrix is then considered as the global optimal solution. The formula for obtaining the solution of the matrix is as follows:

$$x_{id} = r_0(ub_d - lb_d) + lb_d, i = 1, 2, \ldots, N, d = 1, 2, \ldots, Dim \tag{1}$$

where N denotes the total number of gannets. Dim represents the upper limit of the dimension of the solution. lb_d and ub_d represent the upper and lower limits of each dimension. r_0 represents a random number from 0 to 1.

After the initialization of the GOA is completed, Gannets starts to hunt. In the exploration phase, Gannets has two dive modes: one is U-shaped dive, which is suitable for feeding on fish in shallow water and corresponds to Equation (4), the other is a V-shaped dive, which is suitable for feeding on fish in deep water, corresponding to Equation (5),

$$t = 1 - \frac{It_k}{K_{max}}, k = 1, 2, \ldots, K_{max} \tag{2}$$

$$a_u = 2cos(2\pi r_1) \times t \tag{3}$$

$$b_v = 2V(2\pi r_2) \times t \tag{4}$$

$$V_{sh}(y) = \begin{cases} -\dfrac{y}{\pi} + 1 & y \in (0, \pi) \\ \\ \dfrac{y}{\pi} - 1 & y \in (\pi, 2\pi) \end{cases} \tag{5}$$

where It_k denotes the kth iteration and K_{max} denotes the upper limit of the number of iterations, r_1 is a random number from 0 to 1, like r_2.

The probability of choosing these two dive strategies is the same, so a random number q is defined to represent the random selection of hunting strategies. The position update equations are as Equation (6),

$$MX_i(t+1) = \begin{cases} u_1 + u_2 + X_i(t) & q \geq 0.5 \text{ (a)} \\ v_1 + v_2 + X_i(t) & q < 0.5 \text{ (b)} \end{cases} \tag{6}$$

$$u_2 = A(X_i(t) - X_{rand}(t)) \tag{7}$$

$$v_2 = B(X_i(t) - X_{Mean}(t)) \tag{8}$$

$$A = (2r_3 - 1)a_u \tag{9}$$

$$B = (2r_4 - 1)b_v \tag{10}$$

where r_3 and r_4 both range from a random number between 0 and 1, u_1 ranges from $-a_u$ and a_u, and v_1 ranges from $-b_v$ and b_v. The ith solution in the population is denoted by $X_i(t)$. $X_{rand}(t)$ represents a random selection of a solution from the entire population, $X_{Mean}(t)$ represents a solution at the center of the population, and $X_{Mean}(t)$ is calculated as shown Equation (11),

$$X_{Mean}(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{11}$$

During the exploitation phase, when gannets encounter fish that suddenly turn around, they also need to take two actions to develop further. Here, capturing capability is defined as Equation (12),

$$Capturability = \frac{1}{Rt_2} \tag{12}$$

$$t_2 = 1 + \frac{It_k}{K_{max}} \tag{13}$$

$$R = \frac{Mv^2}{L} \tag{14}$$

$$L = 0.2 + (2 - 0.2)r_5 \tag{15}$$

where $M = 2.5$ kg is set by the authors based on the average mass of the Gannet population, r_5 represents a random number from 0 to 1, and $v = 1.5$ m/s represents the speed of the Gannet in the water, given by the authors. If the fish escapes and the location where the fish escapes is within the capture capability of the Gannet, the Gannet will make a position change because it chases the fish; otherwise, the Gannet loses the target and takes a Levy flight for position update to research for the next target x at random, with the position update equations shown in Equation (16),

$$MX_i(It+1) = \begin{cases} X_i(It) + t \times Delt \times (X_i(It) - X_{best}(It)) & Capturability \geq c \text{ (a)} \\ X_{best}(It) - (X_i(It) - X_{best}(It)) \times t \times Lv & Capturability < c \text{ (b)} \end{cases} \tag{16}$$

$$Delt = Capturability \times |X_i(It) - X_{Best}(It)| \tag{17}$$

$$Lv = Levy(Dim) \tag{18}$$

where c is a constant with a value of 0.2. $X_{best}(It)$ denotes the optimal Gannet. $Levy()$ denotes the Levy flight of the Gannet, as shown in Equation (19):

$$Levy(Dim) = 0.01\frac{\mu\sigma}{|v|^{\frac{1}{\beta}}} \tag{19}$$

$$\sigma = \left(\frac{sin\left(\frac{\pi\beta}{2}\right)\Gamma(1+\beta)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta 2^{\left(\frac{\beta-1}{2}\right)}}\right)^{\frac{1}{\beta}} \tag{20}$$

where μ and σ are random values between 0 and 1, and β is a predetermined constant with a value of 1.5.

The above is the formula for updating the position of Gannet during predation. After the above introduction, the pseudocode of the GOA is shown in Algorithm 1:

Algorithm 1: GOA

```
Input: $N_p$: population size; $Dim$: problem dimension; $K_{max}$: The upper limit of the
       number of iterations;
Output: Global optimal individual position in the population; Fitness value;
1  Initialize the position of each Gannet in the population according to Equation (1).
2  Generate a position matrix $MX_i$ based on each initialized Gannet position and
   calculate the fitness value for each Gannet.
3  for $It_k < K_{max}$ do
4      if $rand \geq 0.5$ then
5          for $MX_i$ do
6              if $rand \geq 0.5$ then
7                  Update Gannet $X_i$ via Equation (6a)
8              else
9                  Update Gannet $X_i$ via Equation (6b)
10             end
11         end
12     else
13         for $MX_i$ do
14             if $c \geq 0.2$ then
15                 Update Gannet $X_i$ via Equation (16a)
16             else
17                 Update Gannet $X_i$ via Equation (16b)
18             end
19         end
20     end
21     for $MX_i$ do
22         Calculate fitness value of each Gannet $X_i$ in $MX_i$;
23         Update $MX_i$ based on $X_i$ fitness;
24     end
25 end
```

2.2. Compact Scheme

The compact strategy represents the entire population by using virtual populations, and it has also proven to be effective [27]. When the population is applied with the compact strategy, the update after each iteration is also conducted to the whole population in the form of a virtual population. The entire algorithmic process changes from updating the entire population to updating the probabilistic model representing the population, saving the use of computer memory.

The compact strategy for virtualization of the entire population is to use a probabilistic model. To represent the whole population, a perturbation vector (PV) is used to describe the population [28]. As the algorithm continues, the PV is also changing. The PV is represented as follows: $PV = [\mu^t, \sigma^t]$, where μ denotes the mean value of the PV, σ denotes the standard deviation of the PV, and t is the number of current iterations. In the virtual population, each dimension of all particles of the population corresponds to a Perturbation Vector (PV). Both μ and σ in the PV have corresponding probability density functions (PDF), and the PDF is to be normalized [29]. The distribution of the whole population is represented by the above. After having the PDF corresponding to the population, we can generate the solution x from the PDF. The PDF can be constructed by constructing a Chebyshev polynomial, which leads to a cumulative distribution function (CDF) that takes values ranging from 0 to 1 [30,31]. Because the PDF is defined in $[-1, 1]$, the corresponding CDF needs to be expressed according to the definition of the PDF as follows:

$$CDF = \int_{-1}^{x} PDFdx = \int_{-1}^{x} \frac{\sqrt{\frac{2}{\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sigma(erf(\frac{\mu+1}{\sqrt{2}\sigma}) - erf(\frac{\mu-1}{\sqrt{2}\sigma}))}dx \tag{21}$$

where x takes values ranging from -1 to 1 and erf denotes the error function [32]. The above formula shows that the PDF belongs to a truncated Gaussian distribution, which restricts the distribution function to the interval $[-1, 1]$ and performs a normalization operation. In Equation (22), it can also be transformed into this form as follows:

$$CDF = \frac{erf\left(\frac{\mu+1}{\sqrt{2}\sigma}\right) + erf\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)}{erf\left(\frac{\mu+1}{\sqrt{2}\sigma}\right) - erf\left(\frac{\mu-1}{\sqrt{2}\sigma}\right)} \tag{22}$$

Usually, the generated X_i is updated by the algorithmic update formula for the generated X_i to obtain the new solution X_{new}, and the adaptation is evaluated according to the generated X_i with X_{new}, and the one with good adaptation is called the winner and the one with poor adaptation is called the loser, and then the μ and σ of the PV are updated according to the winner and the loser [33]. The PV update formula is as follows:

$$\mu_i^{t+1} = \mu_i^t + \frac{1}{N_{total}}(winner_i - loser_i) \tag{23}$$

In Equation (23), μ^{t+1} denotes the newly generated mean after iteration. N_{total} denotes the population size. The formula for updating the standard deviation in PV is as follows:

$$\sigma_i^{t+1} = \sqrt{(\sigma_i^t)^2 + (\mu_i^t)^2 - (\mu_{i+1}^t)^2 + \frac{1}{N_{total}}(winner_i^2 - loser_i^2)} \tag{24}$$

The population represented by the probability distribution greatly saves the computer's storage when the algorithm is running. From storing the entire population and updating the entire population at the beginning to storing the PV and updating μ and σ in the PV, it is achieved to run the algorithm with less memory and facilitates the use on resource-constrained devices.

3. Parallel and Compact GOA

In this section, the first part will introduce the proposed parallel communication strategy and the cGOA, and the second part will introduce the parallel and compact hybrid strategy added to the GOA.

3.1. Two Proposed Parallel Communication Strategies and cGOA

The idea of the parallel strategy is to group all particles in equal or unequal fractions, and each group performs its own computation when the algorithm runs [34,35]. We use the grouping of Gannets in the algorithm improvement to improve the convergence and accuracy of the algorithm.

Among the parallel improvements of other algorithms, such as: Parallel Particle Swarm Optimization (PPSO) [36] and Parallel Fish Migration Optimization algorithm with Compact technology (PCFMO) [37], etc. From these algorithms improved by parallel strategies, it can be seen that adding parallel communication to the algorithm is more effective than the original algorithm.

Two novel communication schemes are used in this paper to improve the GOA using the parallel strategy. One parallel communication strategy is to replace the elite solution of a randomly selected group after each iteration when it is better adapted than the elite solution of another randomly selected group, called communication strategy with random replacement; another parallel communication strategy is to replace the elite solution of each group after each iteration when it is better adapted than the elite solution of a randomly selected group, called communication strategy with optimal replacement. In order to better use these two strategies, after each group performs iteration completion, a random selection is used to select a strategy for intergroup communication, and each strategy is selected with a probability of one-half. At the same time, because the two strategies are randomly selected groups and fitness values are calculated for elite solution replacement, a disturbance vector d is added to the elite solution in the group whose communication fails when each group communicates substitution failure, and the original solution is replaced if the disturbed solution is well fitness. Figures 1 and 2 allow us to understand these two parallel communication strategies more intuitively.

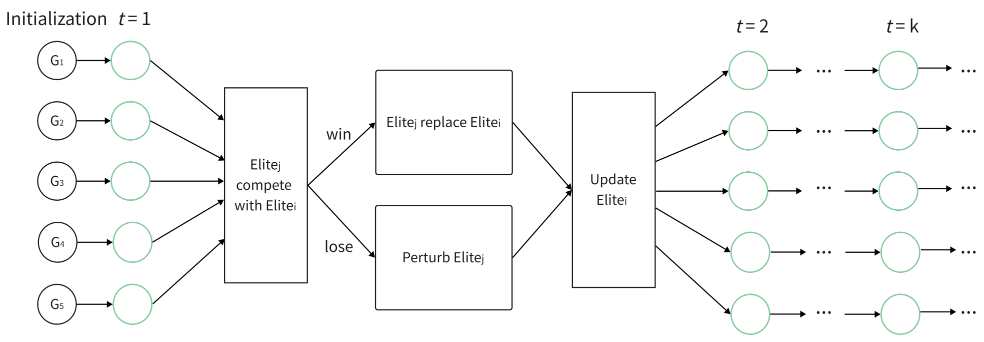


Figure 1. The communication strategy with random replacement.

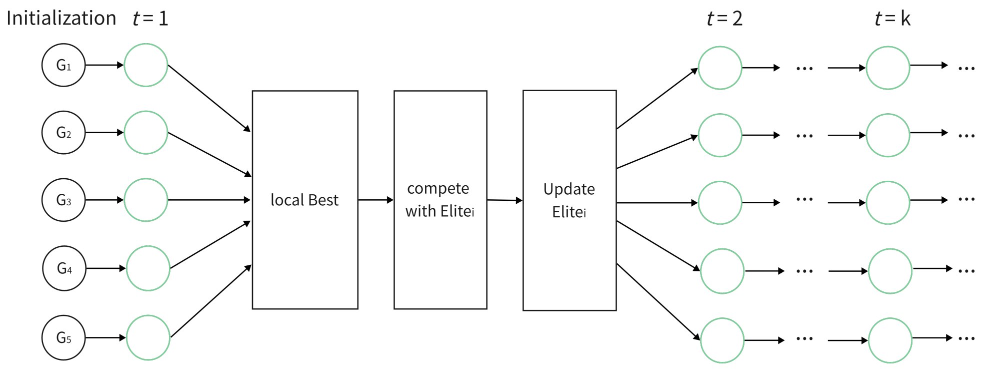


Figure 2. The communication strategy with optimal replacement.

The above is the introduction of the parallel communication strategy used in this paper, and the next will introduce the combination of GOA and compact strategy. In Section 2, we introduced the basic principle of the compact strategy, after which the compact strategy is combined with the GOA, and the combined algorithm is called the cGOA. The cGOA saves the computer storage space occupied by the particles during initialization. The initialization of the GOA is stored for each all particles, while the cGOA is initialized with only one perturbation vector (PV) at the time of initialization. The formula for CDF^{-1} is as follows:

$$y = \sqrt{2}\sigma \times erf^{-1}\left(-erf\left(\frac{\mu+1}{\sqrt{2}\sigma}\right) - x \times erf\left(\frac{\mu-1}{\sqrt{2}\sigma}\right) + x \times erf\left(\frac{\mu+1}{\sqrt{2}\sigma}\right)\right) + \mu \tag{25}$$

In Equation (25), erf^{-1} denotes the inverse function of erf. where y takes values in the range of $[-1, 1]$ and x takes values in the random number of [0, 1]. In order to achieve the mapping of solution y, the following Equation (26) needs to be used to achieve it:

$$y_{ds} = \frac{y}{2}(Ub - Lb) + \frac{1}{2}(Ub + Lb) \tag{26}$$

In Equation (26), Ub and Lb represent the upper and lower limits of each dimension, respectively. y is obtained from Equation (25). y_{ds} is an actual decision solution space. During the iterative process of the cGOA, each iteration is completed using the obtained y_{ds} to update the μ and σ in the PV by Equations (23) and (24) mentioned in Section 2.

3.2. Hibrid Parallel and Compact GOA

Based on our two parallel communication strategies given in the previous section and the compact strategy, a specific implementation of the combination of the parallel and compact strategies is added to the GOA to be improved in this subsection. The improved algorithm after the mixture is called the Parallel Compact Gannet Optimization Algorithm (PCGOA).

In PCGOA, we divided the populations into five groups. While the algorithm is in progress, these five groups perform their own computations. After all five groups complete one iteration, inter-group communication starts. The communication strategy adopted is described in Section 3.1.

In this paper, the join for the compact strategy is to virtualize the populations in each group, because there are five groups, so there are five independent PVs corresponding to the virtual populations in each of the five groups. Each iteration of the algorithm ends by updating the PV corresponding to each of the five groups according to the winner and loser of each of the 5 groups. To facilitate the distinction we take $winner_i(i = 1, 2, 3, 4, 5)$ and $loser_i(i = 1, 2, 3, 4, 5)$ as the winner and loser of the i group. The specific process is as follows:

1. Dividing the entire population into 5 groups and initializing PV_i for each group, where $\sigma_i = 10$, $\mu_i = 0$, $(i = 1, 2, 3, 4, 5)$.
2. Generating the solution X_i via PV_i, generating the corresponding solution X via PV of each group.
3. Compare X and X_{new} of each group, and select the *winner* and *loser* of each group by $[winner, loser] = \text{compete}(X, X_{new})$.
4. X of each group performs the position update formula of the GOA to generate X_{new}.
5. Updating the PV and updating the optimal solution for each group and the global optimal solution according to Equation (16).
6. If the insufficiency condition is met, the algorithm is finalized, otherwise repeat Step 2 to Step 5.

The algorithm flow of the PCGOA is shown in Algorithm 2:

In order to understand PCGOA more clearly, we will analyze the theoretical computational complexity of the PCGOA. From Algorithm 1, we can see that the computational complexity of each iteration of the PCGOA is mainly in lines 9 to 22 of the pseudo-code, and the computational complexity is $O(g \times d)$, where g denotes the total number of groups and d denotes the total number of dimensions. In lines 8 to 24 of the pseudo code, the computational complexity of the whole algorithm is $O(K_{max} \times g \times d)$. In addition, in GOA, the computational complexity of the whole algorithm is $O(K_{max} \times N \times d)$, where N denotes the total number of gannets in the population.

Algorithm 2: PCGOA

```
Input: $N_p$: population size; $Dim$: problem dimension; $K_{max}$:The upper limit of the
       number of iterations;
Output: Global optimal individual position in the population;Fitness value;
1  $t$ = 0; groups = 5;
2  Initialization;
3  for $g$ = 1:groups do
4      Initialize $Group[g].PV$;
5      Generate $Group[g].elite$ via $Group[g].PV$;
6      Calculate the $Group[g].elite$ fitness as $Group[g].fit$
7  end
8  for $It_k<K_{max}$ do
9      for $g$ = 1:groups do
10         generate $Group[g].x$ via $Group[g].PV$;
11         Update $Group[g].x$ by Algorithm 1 to get $Group[g].x_{new}$;
12         $[winner, loser, fit_{winner}] = compete(Group[g].x_{new}, Group[g].elite)$;
13         if $Group[g].x_{new} = winner$ then
14             $Group[g].elite = Groups[g].x_{new}$;
15             Update $Group[g].fit$;
16         end
17         for $i$ = 1:Dim do
18             $temp = (Group[g].\sigma_t[i])^2 + (Group[g].\mu_t[i])^2 - (Group[g].\mu_{t+1}[i])^2$;
19             $Group[g].\mu_{t+1}[i] = Group[g].\mu_t[i] + (winner[i] - loser[i])/N_{total}$;
20             $Group[g].\sigma_{t+1}[i] = \sqrt{temp + (winner[i]^2 - loser[i]^2)/N_{total}}$;
21         end
22     end
23     Update $Fmin$ and $Best$;
24 end
25 for $g = 1 : groups$ do
26     if $rand > 0.5$ then
27         Strategy 1;
28     else
29         Strategy 2;
30     end
31 end
```

4. Experiments

In this section, CEC2013 will be used to test the PCGOA and demonstrate the performance of the PCGOA. A total of 28 functions are covered in CEC2013, including unimodal function distribution (F1–F5), multimodal function (F6–F20) and combinatorial function (F21–F28). These three functions are considered to cover most problems in reality. In the experiments, each algorithm is run in the same environment in CEC2013, which ensures the fairness of the algorithm operation. The development environment used for this experiment is MatLab2018b with Intel(R) Core(TM) I7-10750 H CPU @ 2.60 GHz RAM 16 GB.

4.1. Selection of Comparison Algorithm and Its Parameter Setting

In order to demonstrate the advantages of the PCGOA more comprehensively, the classical algorithms PSO and SCA, the CS algorithm with Levy flight, the MVO algorithm improved by parallel strategy (PMVO), and the AO and BOA algorithms recently proposed in the current research field are selected for experimental comparison in this paper. The specific algorithms and their parameters are set as follows:

- Aquila Optimizer (AO) [38]: $r_0 = 10, delta = 0.1, alpha = 0.1, u = 0.0265$;

- Butterfly Optimization Algorithm (BOA) [39]: $probabibilityswitch = 0.6$, $powerexponent = 0.1, sensorymodality = 0.01$;
- Particle Swarm Optimization (PSO): $V_{max} = 6, V_{min} = 6, c_1 = c_2 = 2, w_3 = 0.3$;
- Sine Cosine Algorithm (SCA) [40]: $a = 2$;
- Parallel Multi-Verse Optimizer (PMVO) [41]: $G = 4, R = 20, 40, \ldots, 2000$, $w = 6, W_{min} = 0.2, W_{max} = 1$;
- Cuckoo Search Algorithm (CS): $P_a = 0.25$.

This paper runs each benchmark function in CEC2013 20 times, each dimension is 30-dimensional, and the functions are evaluated 20,000 times. The advantages of pcGOA over the original algorithm and other algorithms were compared based on the mean and variance of each function run 20 times. See Table 1 for specific data.

Table 1. Simulation results on 30D.

Func_Num		PCGOA	GOA	AO	BOA	PSO	SCA	PMVO	CS
1	Mean	-1.40×10^{3}	-1.40×10^{3}	4.80×10^{3}	5.29×10^{4}	1.54×10^{4}	1.85×10^{4}	**-1.40×10^{3}**	-1.40×10^{3}
	Std	7.28×10^{-2}	7.96×10^{-2}	1.44×10^{3}	5.59×10^{3}	3.19×10^{3}	2.72×10^{3}	5.08×10^{-1}	3.79×10^{-3}
2	Mean	1.88×10^{7}	2.62×10^{7}	1.63×10^{8}	5.19×10^{8}	3.51×10^{8}	2.51×10^{8}	**1.83×10^{7}**	1.37×10^{7}
	Std	4.55×10^{6}	1.07×10^{7}	6.79×10^{7}	4.64×10^{8}	1.24×10^{8}	4.51×10^{7}	3.57×10^{6}	4.14×10^{6}
3	Mean	**5.53×10^{8}**	3.88×10^{9}	3.00×10^{12}	6.73×10^{19}	1.77×10^{14}	1.88×10^{11}	2.58×10^{9}	-1.00×10^{10}
	Std	2.05×10^{9}	7.23×10^{9}	1.19×10^{11}	2.47×10^{20}	4.69×10^{13}	4.65×10^{10}	1.05×10^{9}	7.56×10^{8}
4	Mean	**8.44×10^{3}**	4.76×10^{4}	5.89×10^{4}	5.54×10^{4}	6.72×10^{4}	6.28×10^{4}	4.23×10^{4}	8.58×10^{4}
	Std	2.00×10^{3}	7.26×10^{3}	3.66×10^{3}	2.45×10^{3}	1.10×10^{4}	1.41×10^{4}	1.59×10^{4}	9.94×10^{3}
5	Mean	**-9.99×10^{2}**	-9.88×10^{2}	1.03×10^{2}	3.32×10^{4}	2.41×10^{3}	3.22×10^{3}	-9.00×10^{2}	-1.00×10^{3}
	Std	2.03×10^{-1}	1.37×10	6.18×10^{2}	7.68×10^{3}	1.67×10^{3}	6.30×10^{2}	4.05×10	3.52×10^{-2}
6	Mean	**-8.77×10^{2}**	-7.42×10^{2}	8.60×10	1.29×10^{4}	1.32×10^{3}	1.55×10^{3}	-8.22×10^{2}	-8.63×10^{2}
	Std	2.81×10	3.51×10	2.47×10^{2}	2.60×10^{3}	2.10×10^{3}	4.18×10^{2}	2.13×10	1.75×10
7	Mean	**-6.90×10^{2}**	-6.86×10^{2}	-4.99×10^{2}	9.36×10^{4}	-6.57×10^{2}	-5.95×10^{2}	-6.81×10^{2}	-6.60×10^{2}
	Std	4.05×10	4.28×10	4.67×10^{2}	5.72×10^{5}	7.07×10^{3}	1.14×10^{2}	3.59×10	1.84×10
8	Mean	**-6.79×10^{2}**	-6.79×10^{2}	-6.79×10^{2}	-6.79×10^{2}	-6.79×10^{2}	-6.79×10^{2}	-6.79×10^{2}	-6.79×10^{2}
	Std	4.68×10^{-2}	4.83×10^{-2}	6.52×10^{-2}	4.29×10^{-2}	6.58×10^{-2}	5.31×10^{-2}	6.70×10^{-2}	4.58×10^{-2}
9	Mean	-5.69×10^{2}	-5.65×10^{2}	-5.59×10^{2}	-5.58×10^{2}	-5.63×10^{2}	-5.57×10^{2}	**-5.73×10^{2}**	-5.68×10^{2}
	Std	2.83×10	4.72×10	2.80×10	1.50×10	4.46×10	8.34×10^{-1}	2.77×10	1.19×10
10	Mean	**-4.96×10^{2}**	-2.95×10^{2}	4.59×10^{2}	7.77×10^{3}	2.57×10^{3}	2.31×10^{3}	-4.93×10^{2}	-4.98×10^{2}
	Std	9.91×10^{-1}	3.31×10	3.85×10^{2}	1.10×10^{3}	4.61×10^{2}	4.76×10^{2}	2.39×10	2.13×10^{-1}
11	Mean	-1.80×10^{2}	-2.25×10^{2}	2.46×10	4.95×10^{2}	-1.20×10^{2}	9.51×10	**-3.02×10^{2}**	-2.93×10^{2}
	Std	7.14×10	3.68×10	4.89×10	6.31×10	3.91×10	5.17×10	2.77×10	1.88×10
12	Mean	-2.99×10	-1.65×10^{2}	4.74×10	5.27×10^{2}	1.31×10^{2}	2.10×10^{2}	**-2.20×10^{2}**	-1.16×10^{2}
	Std	1.16×10^{2}	4.40×10	7.30×10	1.01×10^{2}	1.01×10^{2}	4.31×10	4.24×10	2.67×10
13	Mean	4.09×10	**1.48×10**	3.64×10^{2}	6.17×10^{2}	3.43×10^{2}	2.44×10^{2}	2.55×10	1.55×10
	Std	5.15×10	6.85×10	7.71×10	6.01×10	7.76×10	3.17×10	6.13×10	3.24×10
14	Mean	4.07×10^{3}	3.94×10^{3}	5.44×10^{3}	8.29×10^{3}	4.45×10^{3}	7.97×10^{3}	**2.78×10^{3}**	3.41×10^{3}
	Std	5.83×10^{2}	5.53×10^{2}	7.96×10^{2}	3.12×10^{2}	6.54×10^{2}	5.16×10^{2}	5.01×10^{2}	2.32×10^{2}
15	Mean	**4.52×10^{3}**	5.77×10^{3}	5.49×10^{3}	8.05×10^{3}	4.65×10^{3}	8.25×10^{3}	5.52×10^{3}	5.10×10^{3}
	Std	9.84×10^{2}	1.03×10^{3}	7.32×10^{2}	3.44×10^{2}	5.09×10^{2}	4.09×10^{2}	8.36×10^{2}	2.32×10^{2}
16	Mean	**2.01×10^{2}**	2.03×10^{2}	2.03×10^{2}	2.04×10^{2}	2.03×10^{2}	2.04×10^{2}	2.02×10^{2}	2.03×10^{2}
	Std	3.86×10^{-1}	4.08×10^{-1}	4.67×10^{-1}	2.96×10^{-1}	5.61×10^{-1}	4.66×10^{-1}	4.67×10^{-1}	3.69×10^{-1}
17	Mean	5.01×10^{2}	**4.76×10^{2}**	1.00×10^{3}	1.22×10^{3}	6.79×10^{2}	9.61×10^{2}	5.14×10^{2}	4.92×10^{2}
	Std	7.87×10	2.67×10	8.93×10	4.38×10	6.80×10	6.87×10	3.84×10	2.35×10
18	Mean	7.26×10^{2}	6.69×10^{2}	9.85×10^{2}	1.31×10^{3}	8.75×10^{2}	1.10×10^{3}	**6.50×10^{2}**	6.36×10^{2}
	Std	4.86×10	3.98×10	8.54×10	5.94×10	1.10×10^{2}	7.91×10	4.16×10	1.79×10
19	Mean	**5.14×10^{2}**	5.56×10^{2}	8.45×10^{2}	4.61×10^{5}	5.53×10^{4}	2.23×10^{4}	5.15×10^{2}	5.13×10^{2}
	Std	5.16×10	7.21×10	4.45×10^{2}	1.17×10^{5}	1.10×10^{4}	1.61×10^{4}	3.67×10	2.67×10
20	Mean	6.15×10^{2}	**6.13×10^{2}**	6.15×10^{2}	6.15×10^{2}	6.15×10^{2}	6.15×10^{2}	6.15×10^{2}	6.14×10^{2}
	Std	6.75×10^{-1}	1.15×10	1.33×10^{-1}	2.23×10^{-9}	1.33×10^{-1}	3.04×10^{-1}	6.43×10^{-1}	5.62×10^{-1}
21	Mean	**1.01×10^{3}**	1.02×10^{3}	2.49×10^{3}	3.21×10^{3}	2.80×10^{3}	2.88×10^{3}	1.02×10^{3}	9.62×10^{2}
	Std	6.96×10	7.70×10	4.68×10^{2}	5.29×10	1.64×10^{2}	1.11×10^{2}	8.94×10	3.82×10
22	Mean	6.06×10^{3}	**4.53×10^{3}**	7.24×10^{3}	9.61×10^{3}	4.67×10^{3}	8.48×10^{3}	6.51×10^{3}	5.19×10^{3}
	Std	1.07×10^{3}	7.75×10^{2}	1.02×10^{3}	3.29×10^{2}	8.62×10^{2}	3.10×10^{2}	1.35×10^{3}	3.75×10^{2}
23	Mean	7.98×10^{3}	7.14×10^{3}	7.47×10^{3}	9.71×10^{3}	**6.32×10^{3}**	9.11×10^{3}	6.68×10^{3}	6.83×10^{3}
	Std	9.87×10^{2}	6.57×10^{2}	9.50×10^{2}	3.56×10^{2}	1.26×10^{3}	4.26×10^{2}	6.99×10^{2}	3.30×10^{2}
24	Mean	1.28×10^{3}	1.29×10^{3}	1.32×10^{3}	1.45×10^{3}	1.34×10^{3}	1.33×10^{3}	**1.27×10^{3}**	1.30×10^{3}
	Std	1.41×10	1.12×10	9.04×10	2.72×10	3.83×10	4.26×10	8.42×10	4.92×10

Table 1. *Cont.*

Func_Num		PCGOA	GOA	AO	BOA	PSO	SCA	PMVO	CS
25	Mean	1.40×10^3	1.39×10^3	1.43×10^3	1.44×10^3	1.49×10^3	1.43×10^3	$\mathbf{1.37 \times 10^3}$	1.41×10^3
	Std	1.61×10	6.66×10	9.56×10	2.79×10	1.63×10	3.28×10	1.49×10	4.09×10
26	Mean	$\mathbf{1.40 \times 10^3}$	1.40×10^3	1.58×10^3	1.45×10^3	1.58×10^3	1.62×10^3	1.40×10^3	1.40×10^3
	Std	2.89×10^{-1}	5.73×10	7.69×10	7.78×10	9.23×10	7.52×10	7.45×10	6.75×10^{-1}
27	Mean	2.41×10^3	2.52×10^3	2.70×10^3	3.06×10^3	2.46×10^3	2.77×10^3	$\mathbf{2.12 \times 10^3}$	2.43×10^3
	Std	9.90×10	8.95×10	6.41×10	6.26×10	1.27×10^2	4.34×10	1.24×10^2	1.83×10^2
28	Mean	$\mathbf{1.72 \times 10^3}$	1.79×10^3	5.17×10^3	6.12×10^3	4.68×10^3	4.79×10^3	1.74×10^3	1.77×10^3
	Std	1.42×10^3	7.31×10^2	5.55×10^2	2.75×10^2	3.68×10^2	2.30×10^2	5.45×10^2	3.63×10
win/=/los			17/2/9	26/0/2	27/0/1	26/0/2	28/0/0	17/0/11	25/2/1

The bolded data represents the best value taken by this algorithm over other algorithms in the current function. The win, equal sign and los in the lowest row of Table 1 represent the number of wins, ties and failures of the PCGOA in comparison with other algorithms in 28 functions, respectively. We can see that the PCGOA has 17 wins and 2 function ties in the comparison with GOA, and among the winning functions, the PCGOA has an advantage over the GOA in the single-peaked functions as well as the combined functions, and partially in the multimodal functions.

To verify the statistical advantage of the PCGOA and to determine if the PCGOA is significantly different compared to other algorithms, we used the nonparametric Wilcoxon Rank-sum test for this experiment. The significance level was taken to be α of 0.05, and the tested p values are shown in Table 2, where data with p values greater than 0.05 are marked in bold. As can be seen in Table 2, the PCGOA has a significant gap with other algorithms in most cases.

Table 2. *p*-Values of the Wilcoxon rank-sum test for CEC2013 functions.

Function	GOA	AO	BOA	PSO	SCA	PMVO	CS
F1	1.40×10^{-2}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}
F2	3.30×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	**0.1212**	1.83×10^{-4}
F3	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	**0.6776**	6.39×10^{-5}
F4	2.46×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}
F5	1.31×10^{-3}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}
F6	2.20×10^{-3}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	**0.6776**	1.71×10^{-3}
F7	**0.0757**	4.40×10^{-4}	1.83×10^{-4}	1.40×10^{-2}	1.83×10^{-4}	1.73×10^{-2}	**0.5205**
F8	2.20×10^{-3}	1.01×10^{-3}	1.01×10^{-3}	**0.1212**	2.83×10^{-3}	2.46×10^{-4}	3.30×10^{-4}
F9	**0.6776**	1.01×10^{-3}	1.83×10^{-4}	4.52×10^{-2}	1.83×10^{-4}	2.46×10^{-4}	**0.1212**
F10	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	4.40×10^{-4}	1.83×10^{-4}
F11	1.83×10^{-4}	1.73×10^{-2}	1.83×10^{-4}	3.12×10^{-2}	2.46×10^{-4}	1.83×10^{-4}	7.69×10^{-4}
F12	1.31×10^{-3}	2.20×10^{-3}	1.83×10^{-4}	2.20×10^{-3}	2.20×10^{-3}	2.20×10^{-3}	**0.4274**
F13	3.30×10^{-4}	2.46×10^{-4}	1.83×10^{-4}	4.40×10^{-4}	5.83×10^{-4}	3.61×10^{-3}	**0.1041**
F14	2.20×10^{-3}	3.61×10^{-3}	1.83×10^{-4}	**0.3447**	1.83×10^{-4}	**0.4727**	**0.7337**
F15	2.83×10^{-3}	**0.1405**	1.83×10^{-4}	**0.3847**	1.83×10^{-4}	5.80×10^{-3}	2.83×10^{-3}
F16	2.83×10^{-3}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	**0.1405**	1.83×10^{-4}
F17	1.83×10^{-4}	4.40×10^{-4}	1.83×10^{-4}	**0.1620**	1.83×10^{-4}	1.40×10^{-2}	**0.0640**
F18	1.31×10^{-3}	1.83×10^{-4}	1.83×10^{-4}	3.30×10^{-4}	1.83×10^{-4}	**0.2413**	**0.1212**
F19	**0.5205**	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	4.40×10^{-4}	**0.0140**
F20	**0.6764**	1.49×10^{-4}	1.49×10^{-4}	1.73×10^{-4}	7.28×10^{-3}	7.71×10^{-3}	2.83×10^{-3}
F21	1.71×10^{-3}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	**0.1620**	1.83×10^{-4}
F22	1.83×10^{-4}	**0.3075**	1.83×10^{-4}	7.57×10^{-2}	2.46×10^{-4}	**0.2730**	**0.4727**
F23	**0.6776**	1.71×10^{-3}	1.83×10^{-4}	**0.6776**	1.83×10^{-4}	3.12×10^{-2}	**0.1212**
F24	7.28×10^{-3}	3.12×10^{-2}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	**0.9097**
F25	7.69×10^{-4}	7.69×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	3.30×10^{-4}	1.40×10^{-2}
F26	4.40×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	1.83×10^{-4}	7.69×10^{-4}	1.83×10^{-4}
F27	1.40×10^{-2}	1.31×10^{-3}	1.83×10^{-4}	4.40×10^{-4}	1.83×10^{-4}	3.30×10^{-4}	2.11×10^{-2}
F28	**0.0890**	9.11×10^{-3}	1.83×10^{-4}	9.11×10^{-3}	**0.1620**	**0.2365**	**0.6758**

4.2. Convergence Analysis

To better demonstrate the advantages of the PCGOA, the convergence performance of the PCGOA is tested. In this subsection, In this subsection, we present the tests of the PCGOA on 28 benchmark functions of CEC2013 in 10 dimensions. Because the CEC2013 contains three test functions, namely single-peak function, multi-modal function and composite function, the CEC2013 can better test the convergence and robustness of the

PCGOA from many aspects. In order to show the convergence curve of each function more clearly, this paper selects several functions from each function for display.

In Figure 3, the convergence effect of the PCGOA in the single-peaked function is shown. Comparing PCGOA with GOA, PSO, PMVO and other algorithms, the convergence image of the PCGOA in the single-peak function shows that the PCGOA performs well in the single-peak function and converges to the optimal solution faster than several other algorithms. In the stages of single-peaked functions the PCGOA can all converge to the global optimal solution faster than other functions. Because the PCGOA communicates at each iteration, the algorithm performs better with single-peaked functions.

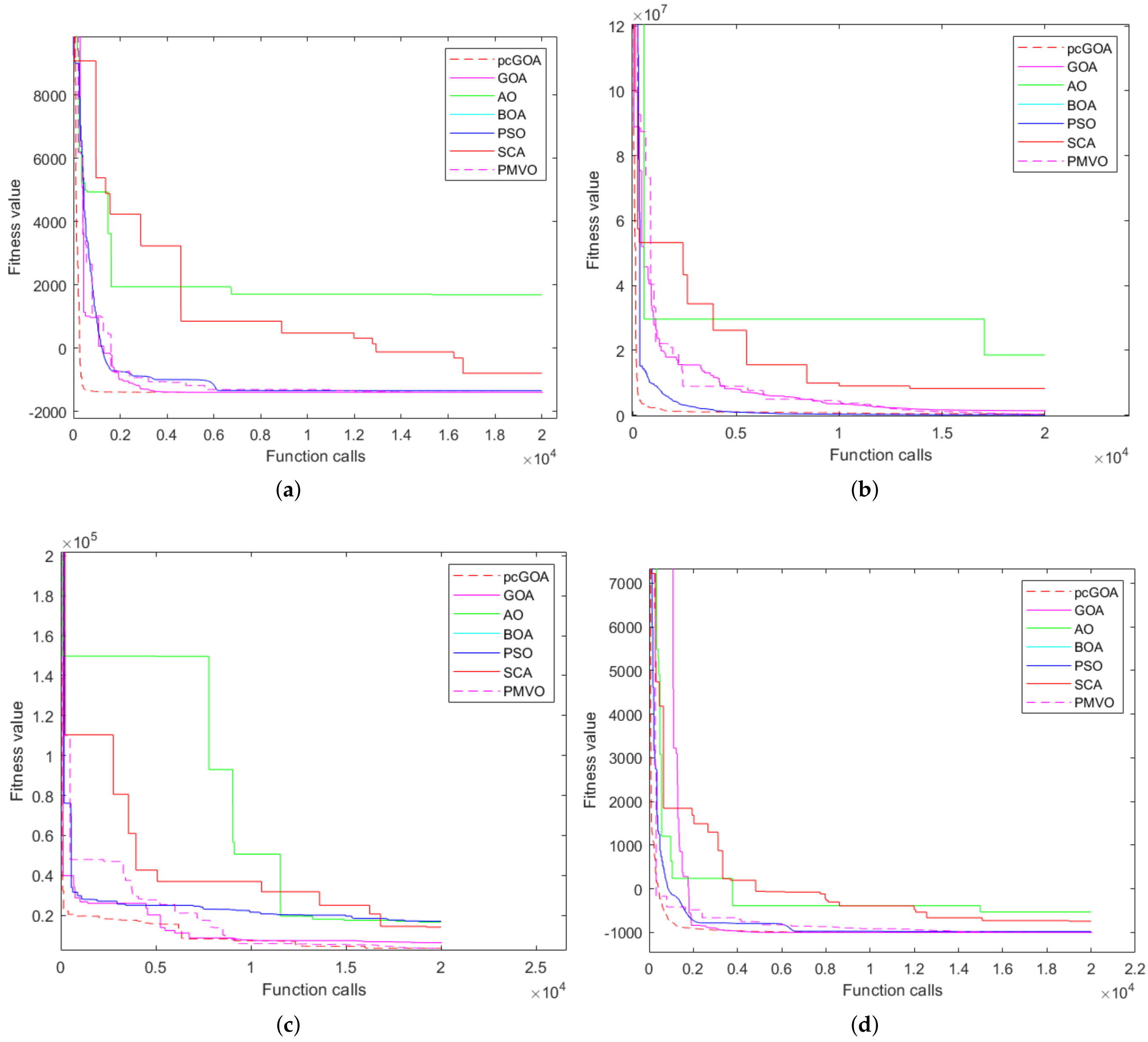
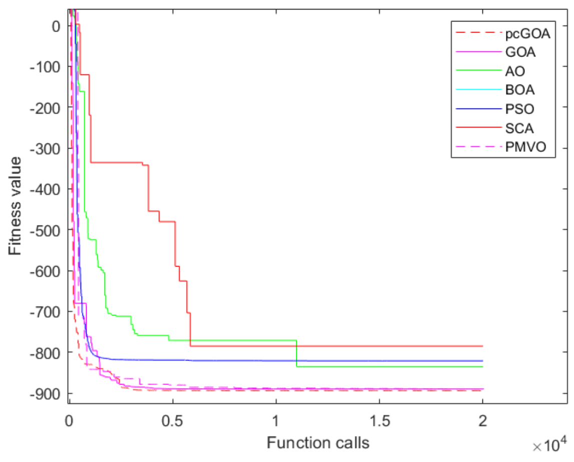
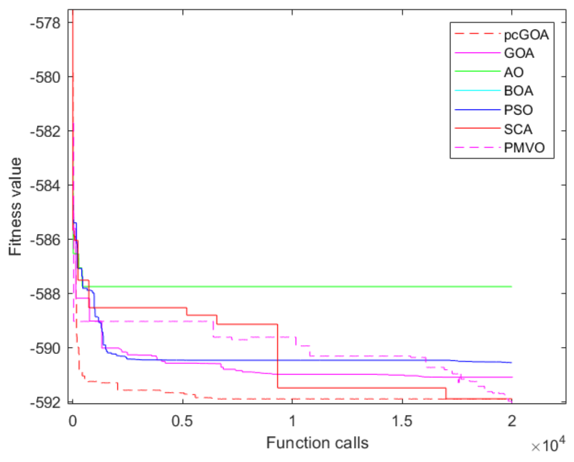


Figure 3. The convergence trend of the unimodal state of the benchmark function in CEC2013. (**a**) F1. (**b**) F2. (**c**) F4. (**d**) F5.

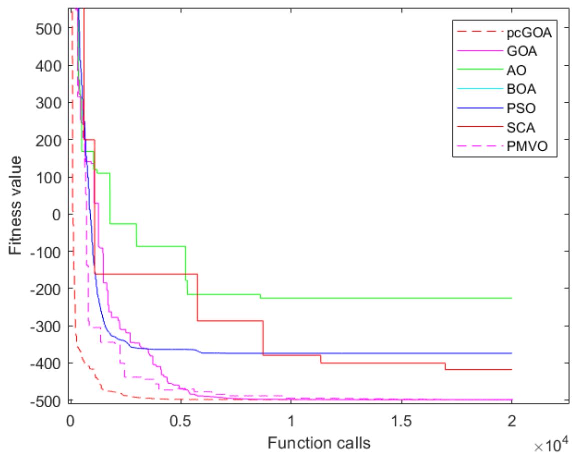
The performance of the PCGOA for multimodal functions is shown in Figure 4. The selected images of functions with more obvious convergence trends show that the convergence of the PCGOA for multimodal functions is also better than other functions, and the convergence to the final results are better than other algorithms. The performance of the PCGOA in the combinatorial function is shown in Figure 5, and four images with more obvious convergence are selected to show in this paper. It can be seen in F21, F23, F24 and F27 that the PCGOA still performs well in the combinatorial functions, and that the PCGOA has good convergence for the same number of evaluation functions. It can be seen from the final results that the PCGOA converges to in most functions that the PCGOA is superior compared to other algorithms.



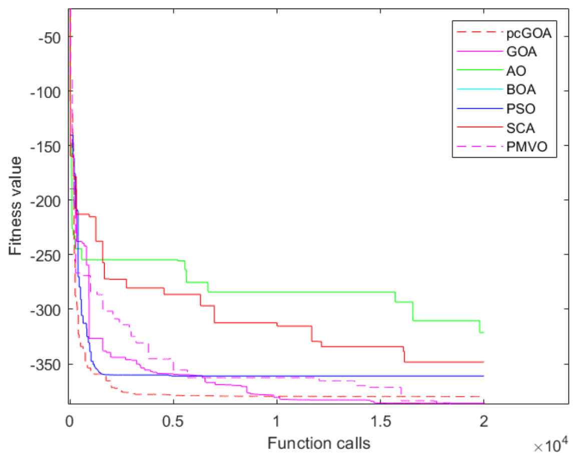
(**a**)



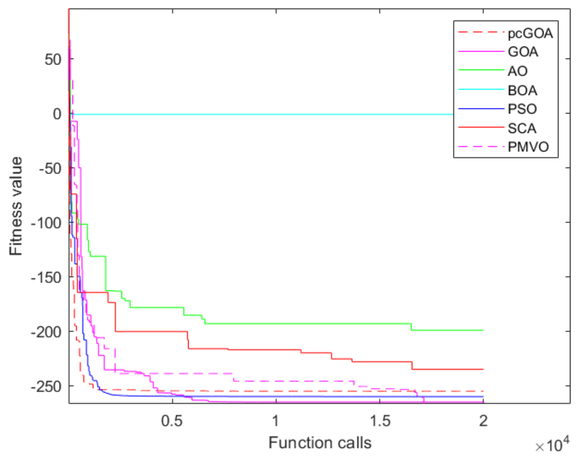
(**b**)



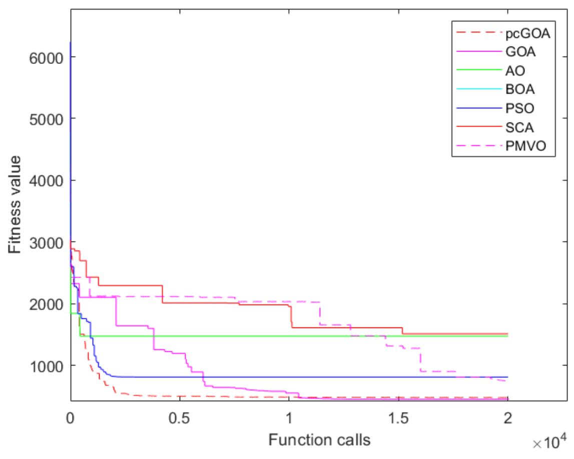
(**c**)



(**d**)

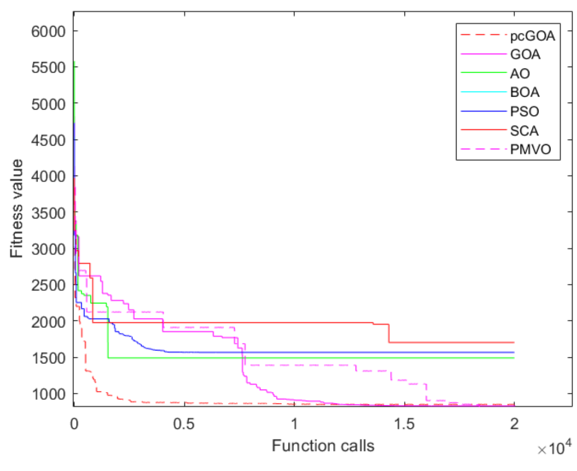


(**e**)

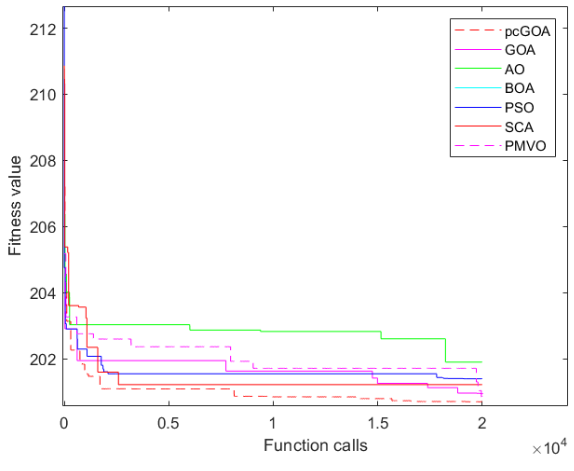


(**f**)

Figure 4. *Cont.*

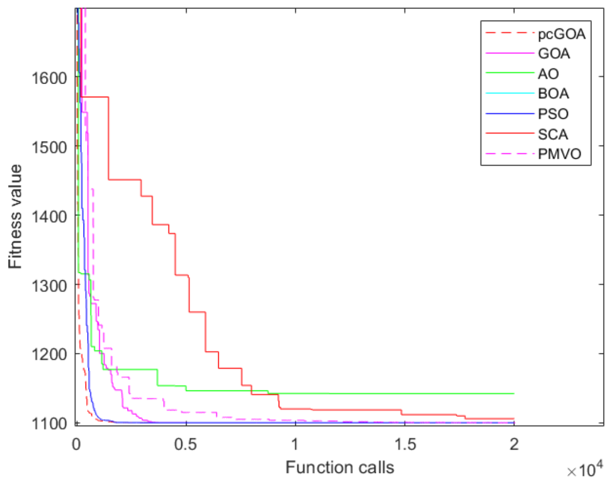


(**g**)

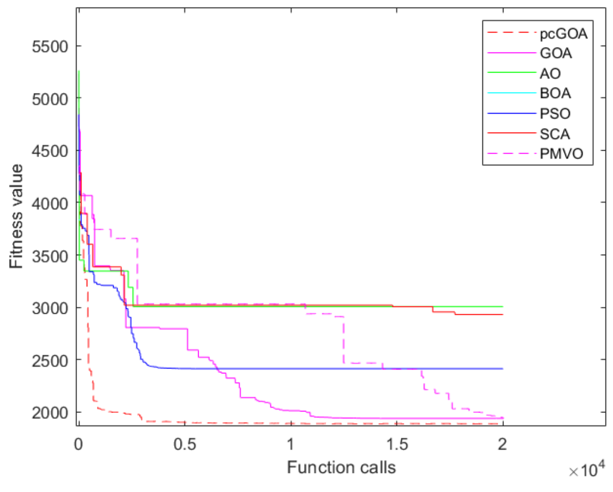


(**h**)

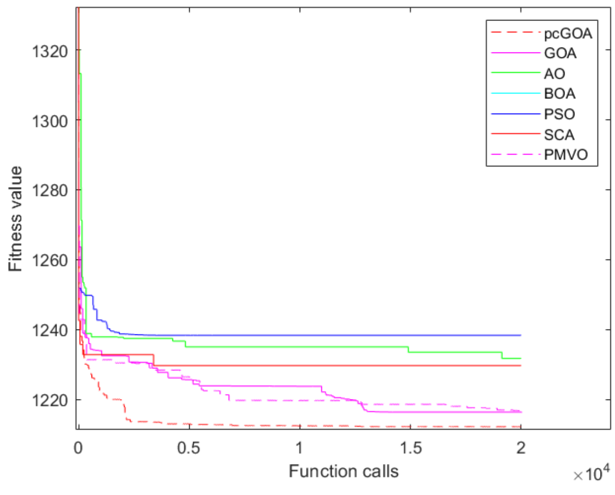
Figure 4. The convergence trend of the multi-modal state of the benchmark function in CEC2013. (**a**) F6. (**b**) F9. (**c**) F10. (**d**) F11. (**e**) F12. (**f**) F14. (**g**) F15. (**h**) F16.



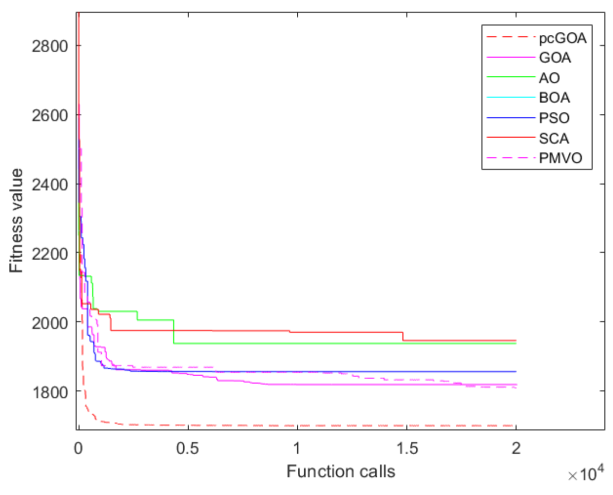
(**a**)



(**b**)



(**c**)



(**d**)

Figure 5. Convergence trend of the composition function of the benchmark function in CEC2013. (**a**) F21. (**b**) F23. (**c**) F24. (**d**) F27.

4.3. Algorithm Memory Analysis

The PCGOA saves computer memory compared to GOA at runtime because the compact strategy used in PCGOA saves memory mainly in terms of storage of the Gannet population in the algorithm run, so this experiment focuses on comparing the size of computer-to-population storage for each iteration.The following table shows the computer memory occupied by several algorithms for the Gannet population at each iteration.

In Table 3, Name is the form in which the algorithm stores the population, Size represents the size stored in the computer, $BytesClass$ indicates the specific size of the storage occupied in the computer, groups represents the number of groups, D represents the total number of dimensions, and N_p indicates the total number of individuals in the population. In the development environment of this experiment, each basic unit is 8 floating-point type data. In $BytesClass$, it is multiplied by 2 because PV of each group is actually a matrix of $2 \times D$.

Table 3. Computer storage of populations at each iteration.

Algorithm	Name	Size	Bytes Class
PCGOA	*Group*	$2 \times groups \times D$	$2 \times groups \times D \times 8$ *double*
GOA	N_p	$N_P \times D$	$N_P \times D \times 8$ *double*
PMVO	*Group*	$N_P \times D$	$N_P \times D \times 8$ *double*
CS	N_p	$N_P \times D$	$N_P \times D \times 8$ *double*

We can see from the above table that the use of the compact strategy achieves the virtualization of the population by storing only a few PVs instead of the whole population, which saves the computer memory when the algorithm is running.

5. Engineering Design Problems

In real life, there will be many optimally solved problems to solve. In this section, the PCGOA is applied to five constrained engineering optimization problems. Tension spring design [42], Pressure vessel design [43], Welded beam design [44], Speed reducer design [45] and Car side impact design [46].

5.1. Constraint Handling

The method of dealing with the boundary constraints in this experiment is the penalty function method, the basic idea of which is to transform the constrained problem into an unconstrained optimization problem with the help of penalty functions, and obtain the solution of the original constrained problem by solving a series of unconstrained optimization problems. This method is used to bring the infeasible point closer to the feasible domain by applying a penalty to it during the iteration. When this point is a feasible point, it is the optimal solution to the original problem.

The constrained optimization problem is transformed into an unconstrained optimization problem by the following equation,

$$\begin{gathered} min : L(X) = f(X) + \sigma \sum_i g(c_i(X)) \\ g(c_i(X)) = max(0, c_i(X))^2 \end{gathered} \tag{27}$$

where i represents the ith constraint, $c_i(X)$ represents a series of constraints, $g(c_i(X))$ is the external penalty function, and σ is the penalty factor. In Equation (27), the value of σ is 1,000,000.

5.2. *Tension Spring Design*

The purpose of pressure vessel design is to minimize its total cost $Func(\vec{\mathbf{X}})$ under its four constraints. There are three design variables involved: the average diameter of the spring coil (x_1), the diameter of the spring wire (x_2), and the number of effective coils of the spring (x_3). The mathematical description is as follows:

$$Func(\vec{\mathbf{X}}) = x_1^2 x_2 (x_3 + 2) \tag{28}$$

The constraints of this engineering optimization problem are as follows:

$$\begin{aligned} g_1(\vec{\mathbf{X}}) &= 1 - \frac{x_2^2 x_3}{7178 x_1^4} \le 0 \\ g_2(\vec{\mathbf{X}}) &= \frac{4x_2^2 - x_1 x_2}{12566 x_1^3 x_2 - x_1^4} + \frac{1}{5108 x_1^2} - 1 \le 0 \\ g_3(\vec{\mathbf{X}}) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0 \\ g_4(\vec{\mathbf{X}}) &= \frac{x_1 + x_2}{1.5} - 1 \le 0 \end{aligned} \tag{29}$$

where the range of values of each variable is as follows:

$$\begin{aligned} 0.05 &\le x_1 \le 2 \\ 0.25 &\le x_2 \le 1.3 \\ 2 &\le x_3 \le 15 \end{aligned} \tag{30}$$

In this paper, the PCGOA is applied to this engineering optimization problem and compared with other algorithms under the same conditions. The optimal solutions derived from each algorithm run for this engineering optimization problem indicate that the PCGOA yields optimal results in solving the problem. The results are shown in Table 4.

Table 4. Comparison results of each algorithm for the tension spring design problem.

Algorithm	x_1	x_2	x_3	*Best*
PCGOA	0.050	0.282	2	**0.00282**
GOA	0.050	0.282	2	**0.00282**
PSO	0.081	0.784	2.0809	0.02120
BOA	0.050	0.282	2	**0.00282**
AO	0.050	0.250	2.8712	0.00305
SCA	0.050	0.282	2	**0.00282**
PMVO	0.050	0.282	2	**0.00282**

5.3. *Pressure Vessel Design*

The purpose of pressure vessel design is to minimize the total cost $Func(\vec{\mathbf{X}})$ while meeting production needs. The total costs affecting pressure vessel design are material, shape and welding. There are four design variables involved: head thickness (x_2), shell thickness (x_1), inner radius (x_3) and vessel length (x_4). The mathematical description is as follows:

$$Func(\vec{\mathbf{X}}) = 3.1661 x_1^2 x_4 + 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 19.84 x_1^2 x_3 \tag{31}$$

The constraints of this engineering optimization problem are as follows:

$$\begin{aligned} g_1(\vec{X}) &= 0.0193x_3 - x_1 \leq 0 \\ g_2(\vec{X}) &= 0.00954x_3 - x_2 \leq 0 \\ g_3(\vec{X}) &= 1296000 - \pi x_3^2 x_4^2 - \frac{4}{3}\pi x_3^3 \leq 0 \\ g_4(\vec{X}) &= x_4 - 240 \leq 0 \end{aligned} \tag{32}$$

where the range of values of each variable is as follows:

$$\begin{aligned} 1 \times 0.0625 &\leq x_1 \\ x_2 &\leq 99 \times 0.0625 \\ 10 &\leq x_3 \\ x_4 &\leq 200 \end{aligned} \tag{33}$$

In this paper, the PCGOA is applied to this engineering optimization problem and compared with other algorithms under the same conditions. The optimal solutions derived by each algorithm running on this engineering optimization problem indicate that the PCGOA has an advantage over the other algorithms in solving the problem. The results are shown in Table 5.

Table 5. Comparison results of each algorithm for the pressure vessel design problem.

Algorithm	x_1	x_2	x_3	x_4	*Best*
PCGOA	0.193	0.096	10.000	64.13	**108.8280**
GOA	0.192	0.095	10.000	64.12	108.8980
PSO	3.399	45.546	19.958	76.36	42,851.7246
BOA	0.192	0.162	10.000	65.967	126.6560
AO	0.194	0.095	10.090	64.537	113.9758
SCA	0.195	0.107	10.000	65.569	114.0628
PMVO	0.192	0.100	10.000	185.174	269.7348

5.4. Welded Beam Design

The purpose of the welded beam design is to minimize its design cost $Func(\vec{X})$ under its seven constraints. There are four design variables involved in the design: the thickness of the weld (x_1), the length of the clamped reinforcement (x_2), the height of the reinforcement (x_3) and the thickness of the reinforcement (x_4). The mathematical description is as follows:

$$Func(\vec{X}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(x_2 + 14) \tag{34}$$

The constraints of this engineering optimization problem are as follows:

$$\begin{aligned}
g_1(\vec{X}) &= \tau(\vec{X}) - 13600 \le 0\\
g_2(\vec{X}) &= \sigma(\vec{X}) - 30000 \le 0\\
g_3(\vec{X}) &= x_1 - x_4 \le 0\\
g_4(\vec{X}) &= 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0\\
g_5(\vec{X}) &= 0.125 - x_1 \le 0\\
g_6(\vec{X}) &= \delta(\vec{X}) - 0.25 \le 0\\
g_7(\vec{X}) &= 6000 - P_a(\vec{X}) \le 0\\
\tau(\vec{X}) &= \sqrt{\tau' + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}\\
\tau' &= \frac{6000}{\sqrt{2}x_1x_2}\\
\tau'' &= \frac{TK}{L}\\
T &= 6000\left(14 + \frac{x_2}{2}\right)\\
K &= \sqrt{\left(\frac{x_1 + x_3}{2}\right)^2 + \frac{x_2^2}{4}}\\
L &= 2\left\{x_1x_2\sqrt{2}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}\\
\sigma(\vec{X}) &= \frac{504000}{x_3^2x_4}\\
P_a(\vec{X}) &= \frac{4.013(30 \times 10^6)\sqrt{\frac{x_3^2x_4^6}{36}}}{196} \times \left(1 - \frac{x_3\sqrt{\frac{30\times10^6}{4(12\times10^6)}}}{28}\right)
\end{aligned} \tag{35}$$

where the range of values of each variable is as follows:

$$\begin{aligned}
0.1 &\le x_1\\
0.1 &\le x_2\\
x_3 &\le 10\\
x_4 &\le 2
\end{aligned} \tag{36}$$

In this paper, the PCGOA is applied to this engineering optimization problem and compared with other algorithms under the same conditions. The optimal solutions derived by each algorithm running on this engineering optimization problem indicate that the PCGOA has an advantage over the other algorithms in solving the problem. The results are shown in Table 6.

Table 6. Comparison results of each algorithm for the welded beam design problem.

Algorithm	x_1	x_2	x_3	x_4	*Best*
PCGOA	0.2056	3.4705	9.0455	0.2057	**1.7258**
GOA	0.2050	3.4305	9.1833	0.2050	1.7380
PSO	0.4193	4.8651	6.6427	0.4279	3.5247
BOA	0.1894	6.7279	7.7389	0.3523	2.9854
AO	0.1656	5.5263	9.1504	0.2052	1.9317
SCA	0.2027	3.8820	8.9529	0.2160	1.8395
PMVO	0.1921	3.7894	9.0467	0.2057	1.7470

5.5. Speed Reducer Design

In this optimization problem, the goal of the reducer design is to minimize its weight $Func(\vec{X})$ under eleven constraints. There are seven design variables involved: tooth width

(x_1), gear module (x_2), number of teeth in the pinion (x_3), length of the first shaft between bearings (x_4), length of the second shaft between bearings (x_5), diameter of the first shaft (x_6) and diameter of the second shaft (x_7). The mathematical description is as follows:

$$\begin{aligned} Func(\vec{X}) =& 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) + 0.7854(x_4x_6^2 + x_5x_7^2) \\ &+ 7.4777(x_6^2 + x_7^2) - 1.508x_1(x_6^2 + x_7^2) \end{aligned} \tag{37}$$

The constraints of this engineering optimization problem are as follows:

$$\begin{aligned} g_1(\vec{X}) &= \frac{27}{x_1x_2^2x_3} - 1 \le 0 \\ g_2(\vec{X}) &= \frac{397.5}{x_1x_2^2x_3^3} - 1 \le 0 \\ g_3(\vec{X}) &= \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0 \\ g_4(\vec{X}) &= \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0 \\ g_5(\vec{X}) &= \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 1.69 \times 10^6} - 1 \le 0 \\ g_6(\vec{X}) &= \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0 \\ g_7(\vec{X}) &= \frac{x_2x_3}{40} - 1 \le 0 \\ g_8(\vec{X}) &= \frac{5x_2}{x_1} - 1 \le 0 \\ g_9(\vec{X}) &= \frac{x_1}{12x_2} - 1 \le 0 \\ g_{10}(\vec{X}) &= \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0 \\ g_{11}(\vec{X}) &= \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0 \end{aligned} \tag{38}$$

where the range of values of each variable is as follows:

$$\begin{aligned} 2.6 \le x_1 \le 3.6 \\ 0.7 \le x_2 \le 0.8 \\ 17.0 \le x_3 \le 28.7 \\ 3.0 \le x_4 \le 8.3 \\ 7.8 \le x_5 \le 8.3 \\ 2.9 \le x_6 \le 3.9 \\ 5 \le x_7 \le 5.5 \end{aligned} \tag{39}$$

In this paper, the PCGOA is applied to this engineering optimization problem and compared with other algorithms under the same conditions. The optimal solutions derived by each algorithm running on this engineering optimization problem indicate that the PCGOA has an advantage over other algorithms. The results are shown in Table 7.

Table 7. Comparison results of each algorithm for the speed reducer design problem.

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	*Best*
PCGOA	3.6	0.8	28	7.3	7.8	3.9	5.2847	**201,613.2**
GOA	3.6	0.8	28	7.3	7.8	3.9	5.2847	**201,613.2**
PSO	3.5524	0.7088	27.7911	7.4979	7.8804	3.7723	5.1926	585,169.9
BOA	3.6	0.8	28	7.3	8.0241	3.9	5.5000	201,760.3
AO	3.6	0.8	28	7.3	8.2965	3.9	5.3078	201,638.6
SCA	3.6	0.8	28	7.3	7.8	3.9	5.2936	201,618.5
PMVO	3.6	0.8	28	7.3	7.9512	3.9	5.2855	201,616.7

5.6. Car Side Impact Design

In daily life, the emergence of automobiles has greatly facilitated people's travel. In this optimization problem, the car will be subjected to side collision, and the purpose of the car side collision design is to minimize the door weight $Func(\vec{\mathbf{X}})$ under ten constraints. There are eleven design variables involved: the thickness of the inner column plate (x_1), the B-pillar reinforcement (x_2), the thickness of the inner floor (x_3), the crossmember (x_4), the door beam (x_5), the door beltline reinforcement (x_6), the roof longitudinal beam (x_7), the inner B-pillar (x_8), the inner floor (x_9), the height of the guardrail (x_{10}) and the material at the crash location (x_{11}). The mathematical description is as follows:

$$Func(\vec{\mathbf{X}}) = 1.98 + 4.90x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7 \tag{40}$$

The constraints of this engineering optimization problem are as follows:

$$\begin{aligned}
g_1(\vec{\mathbf{X}}) =& 1.16 - 0.3717x_2x_4 - 0.00931x_2x_10 - 0.484x_3x_9 + 0.01343x_6x_10 - 1 \leq 0\\
g_2(\vec{\mathbf{X}}) =& 46.36 - 9.9x_2 - 12.9x_1x_2 + 0.1107x_3x_10 - 32 \leq 0\\
g_3(\vec{\mathbf{X}}) =& 33.86 + 2.95x_3 + 0.1792x_3 - 5.057x_1x_2 - 11.0x_2x_8 - 0.0215x_5x_10 - 9.98x_7x_8\\
&+ 22.0x_8x_9 - 32 \leq 0\\
g_4(\vec{\mathbf{X}}) =& 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_10 + 6.63x_6x_9 - 7.7x_7x_8 + 0.32x_9x_{10}\\
&- 32 \leq 0\\
g_5(\vec{\mathbf{X}}) =& 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0144x_3x_5 + 0.0008757x_5x_{10}\\
&+ 0.08045x_6x_9 + 0.00139x_8x_{11} + 0.00001575x_{10}x_{11} - 0.32 \leq 0\\
g_6(\vec{\mathbf{X}}) =& 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.03099x_2x_6 - 0.018x_2x_7\\
&+ 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 + 0.0007715x_5x_{10} - 0.0005354x_6x_{10}\\
&+ 0.00121x_8x_{11} + 0.00184x_9x_{10} - 0.02x_2^2 - 0.32 \leq 0\\
g_7(\vec{\mathbf{X}}) =& 0.74 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 + 0.227x_2^2 - 0.32 \leq 0\\
g_8(\vec{\mathbf{X}}) =& 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} + 0.000191x_{11}^2 - 4 \leq 0\\
g_9(\vec{\mathbf{X}}) =& 10.58 - 0.674x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} + 0.028x_6x_{10}\\
&- 9.9 \leq 0\\
g_{10}(\vec{\mathbf{X}}) =& 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2\\
&- 15.7 \leq 0
\end{aligned} \tag{41}$$

where the range of values of each variable is as follows:

$$\begin{gathered}
0.5 \leq x_1, x_2, x_3, x_4, x_5, x_6, x_7 \leq 1.5\\
x_8, x_9 \in \{0.192, 0.345\}\\
-30 \leq x_{10}, x_{11} \leq 30
\end{gathered} \tag{42}$$

In this paper, the PCGOA is applied to this engineering optimization problem and compared with other algorithms under the same conditions. The optimal solutions derived by each algorithm running on this engineering optimization problem indicate that the PCGOA has an advantage over other algorithms. The results are shown in Table 8.

Table 8. Comparison results of each algorithm for the Car side impact design problem.

	PCGOA	GOA	AO	BOA	PSO	SCA	PMVO
x_1	0.500	0.500	0.514	0.500	0.638	0.500	0.500
x_2	1.001	1.013	0.997	0.926	1.184	0.928	1.056
x_3	0.500	0.500	0.526	0.500	0.618	0.500	0.500
x_4	0.500	0.501	0.532	0.500	0.507	0.645	0.500
x_5	0.500	0.500	0.659	0.681	0.625	0.500	0.507
x_6	1.184	1.436	0.872	0.587	0.987	0.509	0.851
x_7	0.500	0.500	0.500	0.560	0.969	0.500	0.504
x_8	0.192	0.192	0.192	0.192	0.192	0.192	0.192
x_9	0.192	0.192	0.192	0.192	0.192	0.192	0.192
x_{10}	−8.419	−5.597	−12.897	−26.948	−16.715	−30.000	4.384
x_{11}	−0.614	−3.464	−14.472	−12.508	−11.029	−4.106	−1.477
best	**19.074**	19.123	19.755	19.218	23.830	19.266	19.490

6. Conclusions

In this paper, in order to save the use of the GOA in computer memory, the compact strategy adopted achieves the effect of saving memory. The compact strategy is to represent the entire population with a probability model. The combination of parallel strategy and compact strategy allows the algorithm to find the best solution more accurately in various practical problems. In addition, this paper adds a strategy for subpopulation based on parallel communication, which enables each group to achieve better communication. The PCGOA also performs better than the original algorithm in the CEC2013 benchmark function, solving the drawbacks of the original algorithm in terms of large memory consumption and unstable convergence. Finally, this paper applies PCGOA to five engineering optimization problems, and all evidence shows that the test results of the PCGOA are excellent.

Author Contributions: Conceptualization, J.-S.P., B.S., S.-C.C., M.Z. and C.-S.S.; Sotftware, B.S.; formal analysis, B.S. and S.-C.C.; methodology, J.-S.P., B.S., M.Z. and C.-S.S.; writing—original draft, B.S.; writing—review and editing, J.-S.P., B.S. and S.-C.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

References

1. Boussaïd, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [CrossRef]
2. Pan, J.S.; Sun, X.X.; Chu, S.C.; Abraham, A.; Yan, B. Digital watermarking with improved SMS applied for QR code. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104049. [CrossRef]
3. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
4. Bai, Q. Analysis of particle swarm optimization algorithm. *Comput. Inf. Sci.* **2010**, *3*, 180. [CrossRef]
5. Marini, F.; Walczak, B. Particle swarm optimization (PSO). A tutorial. *Chemom. Intell. Lab. Syst.* **2015**, *149*, 153–165. [CrossRef]
6. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
7. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
8. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
9. Dorigo, M.; Di Caro, G. Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Washington, DC, USA, 6–9 July 1999; Volume 2, pp. 1470–1477.
10. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
11. Parpinelli, R.S.; Lopes, H.S.; Freitas, A.A. Data mining with an ant colony optimization algorithm. *IEEE Trans. Evol. Comput.* **2002**, *6*, 321–332. [CrossRef]
12. Yang, X.S. Flower pollination algorithm for global optimization. In Proceedings of the International Conference on Unconventional Computing and Natural Computation, Milan, Italy, 1–5 July 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 240–249.
13. Yang, X.S.; Karamanoglu, M.; He, X. Flower pollination algorithm: A novel approach for multiobjective optimization. *Eng. Optim.* **2014**, *46*, 1222–1237. [CrossRef]

14. Abdel-Basset, M.; Shawky, L.A. Flower pollination algorithm: A comprehensive review. *Artif. Intell. Rev.* **2019**, *52*, 2533–2557. [CrossRef]
15. Song, P.C.; Chu, S.C.; Pan, J.S.; Yang, H. Simplified Phasmatodea population evolution algorithm for optimization. *Complex Intell. Syst.* **2022**, *8*, 2749–2767. [CrossRef]
16. Cheng, M.Y.; Prayogo, D. Symbiotic organisms search: A new metaheuristic optimization algorithm. *Comput. Struct.* **2014**, *139*, 98–112. [CrossRef]
17. Ezugwu, A.E.; Prayogo, D. Symbiotic organisms search algorithm: Theory, recent advances and applications. *Expert Syst. Appl.* **2019**, *119*, 184–209. [CrossRef]
18. Pan, J.S.; Zhang, L.G.; Wang, R.B.; Snášel, V.; Chu, S.C. Gannet Optimization Algorithm: A new metaheuristic algorithm for solving engineering optimization problems. *Math. Comput. Simul.* **2022**, *202*, 343–373. [CrossRef]
19. Pan, J.S.; Liu, N.; Chu, S.C.; Lai, T. An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems. *Inf. Sci.* **2021**, *561*, 304–325. [CrossRef]
20. Chu, S.C.; Du, Z.G.; Peng, Y.J.; Pan, J.S. Fuzzy hierarchical surrogate assists probabilistic particle swarm optimization for expensive high dimensional problem. *Knowl. Based Syst.* **2021**, *220*, 106939. [CrossRef]
21. Xue, Y.; Jiang, J.; Zhao, B.; Ma, T. A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Comput.* **2018**, *22*, 2935–2952. [CrossRef]
22. Mareli, M.; Twala, B. An adaptive Cuckoo search algorithm for optimisation. *Appl. Comput. Inform.* **2018**, *14*, 107–115. [CrossRef]
23. Xue, Y.; Zhu, H.; Liang, J.; Słowik, A. Adaptive crossover operator based multi-objective binary genetic algorithm for feature selection in classification. *Knowl. Based Syst.* **2021**, *227*, 107218. [CrossRef]
24. Harik, G.R.; Lobo, F.G.; Goldberg, D.E. The compact genetic algorithm. *IEEE Trans. Evol. Comput.* **1999**, *3*, 287–297. [CrossRef]
25. Mininno, E.; Neri, F.; Cupertino, F.; Naso, D. Compact differential evolution. *IEEE Trans. Evol. Comput.* **2010**, *15*, 32–54. [CrossRef]
26. Yu, L.; Zheng, Q.; Zhewen, S. Compact Particle Swarm Optimization Algorithm. *J. Xian Jiaotong Univ.* **2006**, *40*, 883.
27. Larra naga, P.; Lozano, J.A. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2001; Volume 2.
28. Nguyen, T.T.; Pan, J.S.; Dao, T.K. A compact bat algorithm for unequal clustering in wireless sensor networks. *Appl. Sci.* **2019**, *9*, 1973. [CrossRef]
29. Bronshtein, I.N.; Semendyayev, K.A. *Handbook of Mathematics*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
30. Mason, J.C.; Handscomb, D.C. *Chebyshev Polynomials*; CRC: Boca Raton, FL, USA, 2002.
31. Cody, W.J. Rational Chebyshev approximations for the error function. *Math. Comput.* **1969**, *23*, 631–637. [CrossRef]
32. Abramowitz, M.; Stegun, I.A. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*; US Government Printing Office: Washington, DC, USA; American Institute of Physics: College Park, MD, USA, 1964; Volume 55.
33. Nguyen, T.T.; Pan, J.S.; Dao, T.K. An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network. *IEEE Access* **2019**, *7*, 75985–75998. [CrossRef]
34. Bäck, T. Parallel optimization of evolutionary algorithms. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Krakov, Poland, 11–15 September 1994; Springer: Berlin/Heidelberg, Germany, 1994; pp. 418–427.
35. Censor, Y.; Zenios, S.A. *Parallel Optimization: Theory, Algorithms, and Applications*; Oxford University Press on Demand: Oxford, UK, 1997.
36. Lalwani, S.; Sharma, H.; Satapathy, S.C.; Deep, K.; Bansal, J.C. A survey on parallel particle swarm optimization algorithms. *Arab. J. Sci. Eng.* **2019**, *44*, 2899–2923. [CrossRef]
37. Chu, S.C.; Xu, X.W.; Yang, S.Y.; Pan, J.S. Parallel fish migration optimization with compact technology based on memory principle for wireless sensor networks. *Knowl. Based Syst.* **2022**, *241*, 108124. [CrossRef]
38. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-Qaness, M.A.; Gandomi, A.H. Aquila optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]
39. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]
40. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
41. Wang, X.; Pan, J.S.; Chu, S.C. A parallel multi-verse optimizer for application in multilevel image segmentation. *IEEE Access* **2020**, *8*, 32018–32030. [CrossRef]
42. Wang, L.; Li, L.-p. An effective differential evolution with level comparison for constrained engineering design. *Struct. Multidiscip. Optim.* **2010**, *41*, 947–963.
43. Shadravan, S.; Naji, H.R.; Bardsiri, V.K. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Eng. Appl. Artif. Intell.* **2019**, *80*, 20–34. [CrossRef]
44. Zhang, Z.; Ding, S.; Jia, W. A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems. *Eng. Appl. Artif. Intell.* **2019**, *85*, 254–268. [CrossRef]

45. Lykouris, T.; Syrgkanis, V.; Tardos, É. Learning and efficiency in games with dynamic population. In Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, Arlington, VI, USA, 10–12 January 2016; pp. 120–129.
46. Marklund, P.O.; Nilsson, L. Optimization of a car body component subjected to side impact. *Struct. Multidiscip. Optim.* **2001**, *21*, 383–392. [CrossRef]

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