Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.056310

ARTICLE

Enhanced Growth Optimizer and Its Application to Multispectral Image Fusion

Jeng-Shyang Pan[1,2], Wenda Li[1], Shu-Chuan Chu[1,*], Xiao Sui[1] and Junzo Watada[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, 413310, Taiwan

[3]Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, 808-0135, Japan

*Corresponding Author: Shu-Chuan Chu. Email: scchu0803@gmail.com

Received: 19 July 2024 Accepted: 15 October 2024 Published: 18 November 2024

ABSTRACT

The growth optimizer (GO) is an innovative and robust metaheuristic optimization algorithm designed to simulate the learning and reflective processes experienced by individuals as they mature within the social environment. However, the original GO algorithm is constrained by two significant limitations: slow convergence and high memory requirements. This restricts its application to large-scale and complex problems. To address these problems, this paper proposes an innovative enhanced growth optimizer (eGO). In contrast to conventional population-based optimization algorithms, the eGO algorithm utilizes a probabilistic model, designated as the virtual population, which is capable of accurately replicating the behavior of actual populations while simultaneously reducing memory consumption. Furthermore, this paper introduces the Lévy flight mechanism, which enhances the diversity and flexibility of the search process, thus further improving the algorithm's global search capability and convergence speed. To verify the effectiveness of the eGO algorithm, a series of experiments were conducted using the CEC2014 and CEC2017 test sets. The results demonstrate that the eGO algorithm outperforms the original GO algorithm and other compact algorithms regarding memory usage and convergence speed, thus exhibiting powerful optimization capabilities. Finally, the eGO algorithm was applied to image fusion. Through a comparative analysis with the existing PSO and GO algorithms and other compact algorithms, the eGO algorithm demonstrates superior performance in image fusion.

KEYWORDS

Growth optimizer; probabilistic model; Lévy flight; image fusion

1 Introduction

As a consequence of the recent advances in computational science, the real-world problems to be solved are becoming increasingly complex [1]. These problems are typically characterized by nonlinearity, large scale, multimodality, and constraints. In response to this, metaheuristic algorithms have emerged as a promising solution to complex optimization challenges.

The growth optimizer (GO) is a population-based meta-heuristic algorithm that models the learning and reflection mechanisms employed by individuals during their social growth. The original

GO algorithm was proposed by Zhang et al. [2] in 2023 for solving continuous and discrete global optimization problems. The search process of the original GO algorithm is divided into two phases: the learning phase and the reflection phase. In the learning phase, which is mainly based on the cooperative search mechanism, each individual learns and acquires knowledge from the knowledge gaps that exist between different individuals. In the reflection phase, individuals use different strategies to recognize and overcome their weaknesses. Growth Optimizer (GO) has demonstrated its effectiveness as an optimization algorithm for the resolution of continuous and discrete global optimization problems [2]. Compared to other metaheuristic methods, GO yielded more encouraging outcomes, particularly concerning solution quality and avoiding local optima. Many studies have demonstrated that GO has the capacity for extensive exploration and the ability to accommodate a wide range of applications. For instance, GO has been effectively employed in the identification of parameters associated with solar photovoltaic cells [3], the segmentation of multilevel threshold images and the deployment of wireless sensor network nodes [4], and the enhancement of intrusion detection systems within the context of IoT and cloud environments [5]. However, very recently, Gao et al. [4] proposed an improved GO algorithm (QAGO) and demonstrated its superiority over the original GO algorithm and some other meta-heuristics in a number of numerical and real-world optimization problems. Gao et al. [4] reported that the original GO has challenges in parameter tuning and operator optimization. In particular, since all hyperparameters of the original GO are fixed values, its performance may be significantly degraded if the parameters are not set properly. Conversely, the search operator structure of the original GO constrains the diversity of the search process. In another study, Fatani et al. [5] proposed an enhanced version of GO, designated MGO, and applied it to an intrusion detection system (IDS). In MGO, the operators of the whale optimization algorithm (WOA) are employed to augment the search process of the original GO. Moreover, Kaveh et al. [6] put forth an enhanced hybrid growth optimizer (IHGO) to address discrete structure optimization challenges, where four enhancements are incorporated into the IHGO relative to the initial GO. Firstly, the learning phase of GO is enhanced through the integration of an improved meta-heuristic exploration phase, namely IAOA, which avoids useless search and enhances exploration. Secondly, the replacement strategy of GO is modified to prevent the loss of the current best solution. Thirdly, the hierarchical structure of GO is modified. Fourthly, the reflection phase of GO is adapted to promote the development of promising regions. GO, one of the traditional optimization algorithms, typically necessitates the storage of a considerable number of solutions and the computation of the corresponding values for each solution. This extensive number of computations presents a challenge to the performance of computing devices, as the large number of computations can lead to conflicts and hinder efficiency. In numerous real-world application scenarios, it is unlikely that high-performance computing devices will be utilized consistently for a multitude of reasons. However, a multitude of optimization problems still necessitate resolution.

Compact algorithms provide an efficient solution by retaining some of the advantages of population-based algorithms without the need to store the entire population in memory. In essence, compact evolutionary algorithms optimize the problem in question using only a solution using the population's statistical features. Consequently, the number of solutions that need to be stored in memory is greatly reduced. As a result, the memory requirements to run these algorithms are greatly reduced compared to population-based algorithms. Compact algorithms fall within the category of Estimation of Distribution Algorithms (EDAs), where the explicit representation of the population is substituted with a probability distribution, as discussed in reference [7]. The initial instantiation of compact algorithms is exemplified by the compact Genetic Algorithm (cGA). The cGA emulates the behavior of a conventional binary-encoded Genetic Algorithm (GA). The findings in Reference [8] reveal that the performance of cGA is nearly comparable to that of GA while significantly

demanding less memory space. The convergence analysis of the cGA is established in Reference [9], while an expanded version, known as the extended compact genetic algorithm (ecGA), is detailed in Reference [10]. Research on the scalability of the ecGA is explored in Reference [11], while Reference [12] documents the utilization of the cGA in training neural networks. Additionally, a number of related algorithmic enhancements utilizing the compact strategy have been put forth, such as compact Bat Algorithm(cBA), compact Particle Swarm Optimization (cPSO) [13], compact Sine Cosine Algorithm(cSCA), etc.

Image fusion remains a cutting-edge topic in the fields of image processing and computer vision exploration. Image fusion refers to the fusion of information from multiple images together to generate a new image, thereby ensuring that the new image contains all the key information and features of the original image. Image fusion techniques can effectively combine image information from different sources to improve the quality and information content of the image. Rahmani et al. introduced an adaptive LHS approach to dynamically calculate band coefficients to achieve more precise detail maps. To improve the accuracy of the spectral representation in non-edge regions, they utilized an edge-holding filter on the Pan image to incorporate the detailed map into each MS band. To enhance the effectiveness of pansharpening, Leung et al. concluded that using Pan images to inject detail maps is not sufficient. To address this, they also employed a linear combination of MS images and Pan images with fixed weights. However, there is a limitation in this approach. The utilization of detail maps with fixed weights may also give rise to distortions, like localized artifacts. To mitigate these impacts, we propose an adaptive framework for the computation of the injected detail maps for each low-resolution (LR) MS band. Different algorithms are used to adaptively compute the weights of the detail maps, and the outcomes are examined and compared. The examination of the experimental findings indicates that the detail map weights obtained adaptively by the eGO algorithm are more appropriate for image fusion compared to other algorithms.

In this paper, we present a novel growth optimizer that employs a compact mechanism and Lévy flight [14] to enhance the optimization and convergence capabilities of the algorithm and reduce its memory requirements. Through the CEC2014 and CEC2017 [15] function sets test, the proposed eGO not only takes less memory but also has higher optimization and convergence capabilities. In comparison to other compact algorithms, the proposed eGO algorithm is also capable of demonstrating considerable competitiveness.

2 Growth Optimizer

The growth optimizer [2] is a robust metaheuristic algorithm that has been designed with inspiration from the learning and reflection mechanisms observed in individuals during their growth processes in society. The process of learning is defined as the acquisition of knowledge from external sources, which enables an individual to grow and develop. Reflection is the process of identifying and addressing the individual's own shortcomings in order to adjust their learning strategies in a way that facilitates their growth. The individual in GO refers to a problem's solution. It is employed to solve both continuous and discrete global optimization problems [16]. The GO algorithm comprises the following steps.

2.1 Initialization

$X_i(0)\,|x_{i,j}(0) \in \left[x_{i,j}^{\min}, x_{i,j}^{\max}\right], i \in [1, N_P], j \in [1, D]$, where $X_i(0)$ represents the *ith* individual of the $0th$ generation, and $x_{i,j}^{\min}$ represents the lower bound of the population, while $x_{i,j}^{\max}$ represents the upper bound of the population. The i denotes the *ith* individual, while the j denotes the *jth* dimension. N_p

represents the size of the population, while D represents the number of dimensions. The initialisation of populations is achieved through the application of the uniform distribution method Eq. (1).

$$x_{i,j}(0) = unifrnd\left(\left[x_{i,j}^{min}, x_{i,j}^{max}\right], Np, D\right) \tag{1}$$

2.2 Learning Phase

The process of identifying and addressing gaps between individuals can significantly contribute to their personal growth. There are four gaps between individuals: The difference between the leader and the elite $\vec{Gap}_1$, the difference between the leader and the lowest-ranked individual $\vec{Gap}_2$, the difference between the elite and the lowest-ranked individual $\vec{Gap}_3$, and the disparity between the characteristics of two randomly selected individuals $\vec{Gap}_4$. The mathematical model is defined by Eq. (2).

$$\begin{aligned} \vec{Gap}_1 &= \vec{x}_{\text{best}} - \vec{x}_{\text{better}} \\ \vec{Gap}_2 &= \vec{x}_{\text{best}} - \vec{x}_{\text{worse}} \\ \vec{Gap}_3 &= \vec{x}_{\text{better}} - \vec{x}_{\text{worse}} \\ \vec{Gap}_4 &= \vec{x}_{\text{L1}} - \vec{x}_{\text{L2}} \end{aligned} \tag{2}$$

where $\vec{x}_{\text{best}}$ signifies the society leader, while $\vec{x}_{\text{better}}$ signifies one of the next $P_1 - 1$ best individuals, who are called an elite. In the context of GO, the leader and the elites collectively constitute the top tier of the society. $\vec{x}_{\text{worse}}$ denotes an individual from the population who is one of the P_1 individuals and is positioned at the lowest level of the social class hierarchy. Both $\vec{x}_{\text{L1}}$ and $\vec{x}_{\text{L2}}$ are randomly chosen individuals distinct from *ith* individual. $\vec{Gap}_k$ $(k = 1, 2, 3, 4)$ represents the distinction between two individuals. $\vec{Gap}_k$ enables learners to comprehensively grasp the dissimilarities between two individuals to their advantage.

In order to reflect the aforementioned variability, a learning factor LF_k is introduced in each of the four disparity measures. For the *ith* individual, LF_k affects its learning effect on the *kth* set of disparities. LF_k is therefore modeled as follows Eq. (3).

$$LF_k = \frac{||\vec{Gap}_k||}{\Sigma_{k=1}^{4}||\vec{Gap}_k||} \tag{3}$$

where $LF_k \in [0, 1]$ represents the Euclidean distance's normalized ratio for $\vec{Gap}_k$. As $\vec{Gap}_k$ increases, LF_k also increases, indicating that the *ith* individual will acquire more knowledge from the *kth* gap.

The *ith* individual employs SF_i to gauge the extent of knowledge deemed acceptable for personal improvement, as illustrated in (Eq. (4)). A higher SF_i indicates that individual SF_i needs to acquire additional knowledge to enhance itself.

$$SF_i = \frac{GR_i}{GR_{max}} \tag{4}$$

Regarding GR_i, individual i assimilates certain knowledge from them, forming the *kth* group of knowledge acquisition ($\vec{KA}_k$). For individual i, $\vec{KA}_k$ results from the application of LF_k and SF_i on the *kth* group gap, and LF_k is modeled as follows Eq. (5).

$$\vec{KA}_k = SF_i \cdot LF_k \cdot \vec{Gap}_k \tag{5}$$

where $\vec{KA}_k$ is the knowledge acquired by the *ith* individual from $\vec{Gap}_k$. SF_i reflects an evaluation of its personal circumstances, while LF_k signifies an appraisal of the external conditions.

Through assimilating knowledge disparities among various individuals, the *ith* individual undergoes a comprehensive process of accumulating substantial knowledge. The *ith* the individual's specific learning process is shown in Eq. (6).

$$\vec{x}_i^{It+1} = \vec{x}_i^{It} + K\vec{A}_1 + K\vec{A}_2 + K\vec{A}_3 + K\vec{A}_4 \tag{6}$$

where *It* represents the current iteration count, and $\vec{x}_i$ signifies the *ith* individual who incorporates the acquired knowledge from the learning phase to facilitate personal growth.

After the phase of learning adjustment, the quality of each individual may either improve or deteriorate. Therefore, it's crucial to verify whether genuine progress has been achieved. If progress is observed, the growth resistance (GR_i) of the individual decreases, and their rank increases. In the event that the *ith* individual degenerates, there is a high probability that the *ith* individual will relinquish part of the learned knowledge. Nevertheless, it is conceivable that the acquired knowledge may be retained with a minimal probability, given that the learning process necessitates a considerable investment of time and effort on the *ith* individual. In this context, P_2 is accountable for regulating the probability of retention. The process is detailed by Eq. (7).

$$\vec{x}_i^{It+1} = \begin{cases} \vec{x}_i^{It+1} & if \quad f\left(\vec{x}_i^{It+1}\right) < f\left(\vec{x}_i^{It}\right) \\ \begin{cases} \vec{x}_i^{It+1} & if \quad r_1 < P_2 \\ \vec{x}_i^{It} & else \end{cases} & else \end{cases} \tag{7}$$

where $r_1 \in [0,\ 1]$ represents a uniformly distributed random number, *ind* (*i*) denotes the *ith* individual's ranking is determined by the ascending order of *GR*, and P_2 indicates whether, in the event that the *ith* individual neglects to update, the just learned knowledge is kept. In this case, P_2 is equal to 0.001. The whole conditional judgment statement for maintaining newly acquired information (in case of an individual update failure) is included below, because to the limited space in Eq. (7): $r_1 < P_2$ and $ind\,(i) \sim = ind\,(1)$. This indicates that in the event of a failed individual update, there is a 0.001 the probability that the individual will be included in the subsequent generation of the population. Furthermore, it makes sure that the present global optimal individual can not be changed, since doing so could cause the algorithm to converge incorrectly.

2.3 Reflection Phase

The processes of learning and reflection are mutually reinforcing. Individuals should develop both learning and reflection skills. Individuals should examine for and correct inadequacies in all areas, and retain information. To make up for deficiencies, they can learn from successful persons while retaining their strengths. If a lesson cannot be repaired, it is recommended to forgo past information and re-learn systematically. Three distinct processing methods exist. The initial approach is to maintain the original dimension; the subsequent way involves selecting an upper-class individual to direct the *jth* dimension of the *ith* individual, and the third method entails reconstructing the *jth* dimension of an individual with a low probability based on the second method. Mathematical modeling of GO's reflection process through Eqs. (8) and (9).

$$\vec{x}_{i,j}^{It+1} = \begin{cases} \begin{cases} lb + r_4 \times (ub - lb) & if \quad r_3 < AF \\ \vec{x}_{i,j}^{It} + r_5 \times (R_j - \vec{x}_{i,j}^{It}) & else \end{cases} & if \quad r_2 < P_3 \\ \vec{x}_{i,j}^{It} & else \end{cases} \tag{8}$$

$$AF = 0.01 + 0.99 \times \left(1 - \frac{FEs}{MaxFEs}\right) \tag{9}$$

In Eq. (8), the upper and lower bounds of the search domain are represented by *ub* and *lb*, respectively. The values of r_2, r_3, r_4, and r_5 are uniformly distributed random numbers within the range of [0, 1]. The probability of reflections is controlled by P_3, which is typically set to 0.3. In Eq. (9), the attenuation factor (AF) is calculated based on the current number of evaluations (FEs) and the maximum number of evaluations ($MaxFEs$). During the iteration of the procedure, the value of AF gradually approaches 0.01. This indicates that as individuals make progress, they are able to avoid spending excessive time on frequent initializations. The attenuation factor (AF) plays a crucial role in regulating the balance between exploration and exploitation during the optimization process. Initial phase (high value of AF): At the beginning of the algorithm, the attenuation factor AF is close to 1. This means that the positions of the individuals are more often randomly initialized. The purpose of this random initialization is to improve the exploration of the search space and help the algorithm avoid falling into local optimal solutions too early. Later stages (smaller AF values): As the algorithm runs, the current number of evaluations (FEs) gradually approaches the maximum number of evaluations ($MaxFEs$), and the attenuation factor (AF) gradually decreases to 0.01. At this stage, the algorithm is more biased towards exploitation, i.e., optimizing and improving existing solutions rather than exploring new regions. During the phase of reflection, the *jth* dimension of the *ith* individual is influenced by an upper-level individual ($\vec{R}$). $\vec{R}$ represents the higher-level individual, which is the current reflective learning guide for individual $\vec{x}_i$. R_j presents the knowledge of the $\vec{R}$ dimension of the $\vec{R}$ vector. The range of vector $\vec{R}$ is determined by selecting the top $P_1 + 1$ individuals, who are considered leaders or elites, from the population. This implies that when individual $\vec{x}_i$ needs to learn from others, the *jth* dimension of their learning will come from the other individuals, influenced by the upper-level individual $\vec{R}$.

3 Enhanced Growth Optimizer

It would appear that the extant literature does not endeavor to put forth a mechanism that enhances GO in compact strategy, thereby reducing the algorithm's memory requirements to a considerable extent. GO may potentially become trapped in local optima and exhibit a slow convergence when attempting to solve high-dimensional, complex optimization problems. In response to the aforementioned questions, this section outlines the methodology for modifying the GO algorithm using the compact strategy, as well as subsequent enhancements to the compact strategy.

3.1 Compact Strategy

The core objective of implementing a compact strategy is to decrease memory consumption while maintaining, or potentially enhancing, the performance of the original algorithm. Enhanced growth optimizer (eGO) is a framework grounded in a growth optimizer which incorporates concepts and designs from GO algorithms. In this section, we will delve into further details regarding the eGO algorithm. The objective of eGO is to emulate the operational characteristics of the underlying GO algorithm while reducing the consumption of memory resources. A virtual population is used to convert the underlying GO individuals into a concise algorithm. The virtual population can be thought of as a probabilistic model of the solution population. It is stored in a data structure called the perturbation vector (PV). This approach permits the efficient representation and processing of potential solutions without the necessity of maintaining an actual population of individuals. In this

framework, the perturbation vector captures the statistical properties of the population, defining the distribution of solutions by encoding the mean, variance, or other relevant parameters. We utilize perturbation vector (PV) to generate new candidate solutions through perturbation.

$$PV^t = [\mu^t, \sigma^t] \tag{10}$$

In the Eq. (10), μ^t and σ^t represent two parameters corresponding to the mean and standard deviation of the PV vector. The parameter t signifies the current iteration count. The values of μ^t and σ^t are limited to the range of probability density functions (PDF) [17], typically adjusted to fall within the interval of [−1, 1]. The PDF's magnitude is normalized to maintain an area of 1, ensuring a uniform distribution with a complete shape, which aids in achieving adequate convergence.

The process of virtual population initialization is as follows: Each parameter is set such that $\mu^t = 0$ and $\sigma^t = \lambda$, where λ represents a large constant (in this paper, we set λ =10). This initial setting ensures the establishment of a normal distribution characterized by truncated and broad shapes.

It takes a thorough explanation to understand the sampling mechanism of a individual $\vec{x}_i$ associated with a generic candidate individual x from PV. A truncated Gaussian PDF with mean value μ and standard deviation σ is linked to each individual indexed by i. Eq. (11) describes the PDF.

$$PDF = \frac{e^{-\frac{(x-\mu[i])^2}{2\times\sigma[i]^2}} \times \sqrt{\frac{2}{\pi}}}{\sigma\,[i] \times \left(erf\left(\frac{\mu\,[i]+1}{\sqrt{2}\times\sigma\,[i]} - erf\left(\frac{\mu\,[i]-1}{\sqrt{2}\times\sigma\,[i]} \right) \right) \right)} \tag{11}$$

The PDF represents the probability distribution function of PV, with parameters μ and σ being associated with a truncated Gaussian distribution. A novel candidate individual is generated through a process of iterative directing towards a favorable area of the most optimal individual. The PV components are updated by learning from previous generations. The error function erf, as defined in Reference [13], plays a crucial role. The PDF is related to the cumulative distribution function (CDF), constructed using a Chebyshev polynomial [18]. The CDF represents a random individual x that takes on real values and follows a probability distribution. In that case, the observed value is less than or equal to $\vec{x}_i^t$. The connection between CDF and PDF is established as follows $CDF = \int_0^1 PDF\,(x)\,dx$. Operations on PV involve sampling the design variable by producing random values within the interval of (0, 1).

During the iterative process of the compact algorithm, this paper introduces a comparative function based on two parameters to facilitate the identification of superior individuals. The two modifiable parameters, pre-learning and post-learning, correspond to two sample individuals within the PV operation. The individual represented by the *winner* attains a higher fitness function value compared to another virtual members, while the individual represented by the *loser* falls short of meeting the fitness evaluation criterion. These two individuals, *winner* and *loser*, are derived from the objective function calculation and contribute to generating a new candidate individual. This new individual is then compared against the original global optimal individual, resulting in the identification of new *winner* and *loser*, When updating PV, adjustments to μ and σ can be made based on the following guidelines: If $\mu = 1$, the update rule for every element within it, μ^{t+1} and σ^{t+1}, is as follows (12) and (13):

$$\mu_i^{t+1} = \mu_i^t + \frac{winner_i - loser_i}{N_p} \tag{12}$$

The virtual population size is denoted by N_P, and the value of σ is specified accordingly. The Eq. (13) provides the updating rule for each element.

$$\sigma_i^{t+1} = \sqrt{\left(\sigma_i^t\right)^2 + \left(\mu_i^{t+1}\right)^2 + \frac{winner_i^2 - loser_i^2}{N_p}} \tag{13}$$

It should be remembered that N_p is an empirical value that does not match the real population size, which is typically many times larger.

The proposed eGO utilizes a perturbation $PV^t = [\mu^t, \sigma^t]$ with a structure identical to that presented in Eq. (10). At the commencement of the optimization process, the initialization of the perturbation vector (PV) is conducted according to the procedure outlined in reference [8]. Individuals need to learn from the disparities among leaders, elites, and inferior performers; therefore, it is necessary to generate three individuals using PV and identify leaders, elites, and inferior performers by comparing their fitness values.

The Eq. (14) for assimilating the knowledge gap among different individuals is modeled after the GO formula.

$$\vec{x}^{It+1} = \vec{x}^{It} + K\vec{A}_1 + K\vec{A}_2 + K\vec{A}_3 + K\vec{A}_4 \tag{14}$$

What can be observed is that the Eq. (14) for assimilating the knowledge gap among different individuals shares the same structure as the Eq. (6) in GO. However, while index i seems to indicate the particle under investigation in the population-based GO, index i does not appear in its compact version.

Similarly, the other two Eqs. (15) and (16) used in the learning and reflection phases are replicated in the same manner.

$$\vec{x}^{It+1} = \begin{cases} \vec{x}^{It+1} & if \quad f\left(\vec{x}^{It+1}\right) < f\left(\vec{x}_i^{It}\right) \\ \begin{cases} \vec{x}^{It+1} & if \quad r_1 < P_2 \\ \vec{x}^{It} & else \end{cases} & else \end{cases} \tag{15}$$

$$\vec{x}_j^{It+1} = \begin{cases} \begin{cases} lb + r_4 \times (ub - lb) & if \quad r_3 < AF \\ \vec{x}_j^{It} + r_5 \times \left(R_j - \vec{x}_j^{It}\right) & else \end{cases} & if \quad r_2 < P_3 \\ \vec{x}_j^{It} & else \end{cases} \tag{16}$$

In eGO, the compact strategy reduces memory usage by employing a probabilistic model representation, which is more memory efficient than the original GO. Traditional algorithms, such as GO, rely on maintaining a complete population, wherein each individual is required to store information such as their position, speed, fitness value, and so forth. For an optimization problem with problem dimension D, the size of the population is N_p, and the memory footprint is $O\left(N_p \times D\right)$. eGO, on the other hand, uses a probability distribution to show the information in the population. This means that only the probability values of three decision variables need to be stored, which cuts the memory overhead by three times D. This compact representation obviates the need to store specific individuals or populations, which markedly reduces memory usage and is particularly well-suited to memory-constrained scenarios.

3.2 Lévy Flight

Lévy flight [14] represents a specific type of stochastic wandering with a probability distribution of step sizes proposed by the French mathematician Levy. The introduction of Lévy flight results in random movements with longer jumps and multiple abrupt changes in direction [19]. This implies that short local searches are interspersed with long jump global searches. In the algorithm space, this kind of flight finds farther-off solutions while improving the neighborhood search close to the ideal answer. This strategy can effectively increase the diversity of individuals and help the algorithm with escaping the local optimal solution [20]. Lévy flights introduce a probability distribution of step lengths, predominantly short steps but with occasional long jumps. This property is well suited for exploring unexplored regions of the search space. Long jumps allow the algorithm to jump out of the local optimal trap and explore more distant regions, increasing the chance of discovering a globally optimal solution. While long jumps make global exploration more efficient, short step sizes are used for local searches near the current optimal solution, which helps to refine the solution and thus speeds up convergence to the optimal solution. Many optimization algorithms are prone to falling into a local optimum, especially for complex multi-peaked functions. Long jumps at Lévy flight reduce this risk. These jumps break the algorithm's stagnation in a locally optimal solution, allowing it to jump out of the local optimum and continue the global search. Such episodic long jumps disrupt the algorithm's tendency to converge prematurely on a locally optimal solution, providing new opportunities for further search. In some cases, especially in complex search spaces with multiple local optima, random long jumps in Lévy's flight can bring the algorithm directly close to the global optimum. This greatly speeds up the convergence process, as the algorithm is able to skip large regions of inefficiency. Once the algorithm has quickly reached the more optimal region, it can switch to fine-grained local search, further increasing the speed of convergence. Therefore, in this thesis, we consider adding Lévy flight to the end of the reflection phase of the algorithm and using a greedy strategy to select the new individuals obtained through Lévy flight and the individuals after the reflection phase.

The flowchart of the enhanced growth optimizer is displayed in Fig. 1.

The pseudocode of enhanced growth optimizer is displayed in Algorithm 1.

Algorithm 1: Pseudocode of enhanced growth optimizer

Input: N_p: the virtual population size; D: problem dimension; [lb, ub]: problem boundary; $P_2 = 0.001$; $P_3 = 0.3$; FEs=0;
Output: Global optimal solution;
1: //PV Initialization;
2: for $i : n$ do
3: Initialize $\mu[i] = 0$
4: Initialize $\sigma[i] = 10$
5: **end for**
6: generate xt by PV
7: while termination condition is not satisfied **do**
8: generate three sample random individuals(xm) from PV;
9: Evaluate fitness of each point in xm using fhd and Sort xm based on fitness;
10: $Best_X$ = xm with the best fitness;
11: Sample WorstX and BetterX from xm;
12: generate two random individuals that are different from xm and xt;
13: Store the unupdated xt as xta;
14: % Learning phase:

(Continued)

Algorithm 1 (continued)

Calculate $\vec{Gap}_k$(k = 1,2,3,4);
Calculate $\boldsymbol{LF}_k$(k = 1,2,3,4);
Calculate SF;
Calculate $\vec{\boldsymbol{KA}}_k$ (k = 1,2,3,4);
Execute the learning process for the specific xt individual in accordance with Eq. (14);
Complete updating the individual xt based on Eq. (15);
Update gbestX and gbestfitness if necessary;
Update xm with newxt and re-evaluate fitness;
% Reflection phase:
Execute the reflection process for the individual specific xt in accordance with according to Eqs. (16) and (9);
Complete updating the individual xt based on Eq. (15);
Generate lxt using Lévy flight by xt;
Complete the update of the individual xt by evaluating lxt and xt fitness;
[winner,loser] = compete(xt,xta);
Update mu and sigma by Eqs. (12) and (13)
Update gbestX and gbestfitness if necessary;
FEs = FEs + 1;
end while

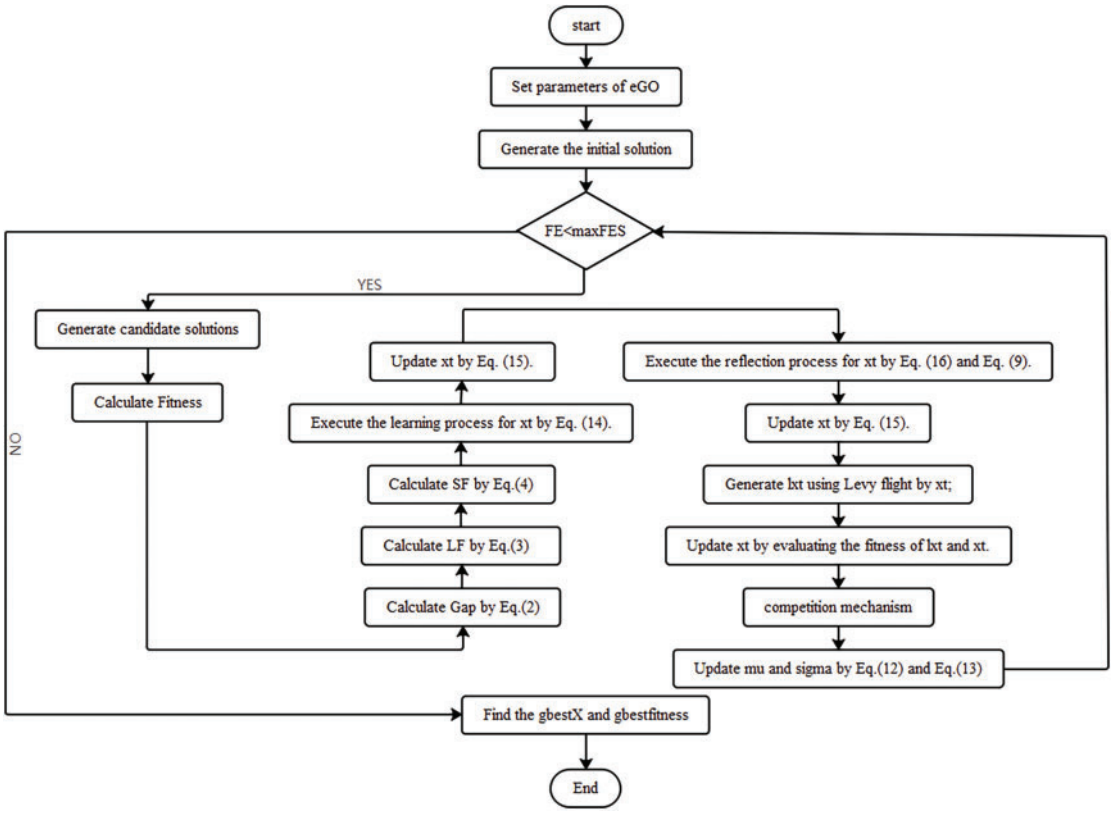


Figure 1: Flowchart of enhanced growth optimizer

4 Numerical Experimental Results and Analysis

To evaluate the performance of the proposed algorithm, we tested eGO using 30 test functions from CEC2014 and CEC2017. This paper compares eGO with eGO_l (the ehanced GO algorithm without Lévy flight), GO and other commonly used compact algorithms, including compact Cuckoo Search algorithm (cCS) [21], compact Pigeon-Inspired Optimization (cPIO) [22] and the above three compact algorithms cPSO, cBA and cSCA. CEC2014 and CEC2017 comprise various types of test functions, including single-objective optimization functions, multi-objective optimization functions, and constrained optimization functions, which can address a wide range of optimization problems. The included test functions have been meticulously designed and validated to ensure a fair evaluation of various optimization algorithms.

4.1 Experimental Parameter Settings

This paper uses the maximum number of evaluations (*MaxFEs*) as the termination criterion. The conditions for each algorithm to obtain relevant data include 60 benchmark problems, 30/50/100 dimensions, and each algorithm runs 20 times independently. The compact algorithms in the experiments were initialized using the same PV, while the original GO used random initialization in the search space. The size of the virtual population was analyzed in order to ascertain the sensitivity of the algorithm. The results of this analysis are presented in Table 1, which demonstrates that the algorithm performs similarly in terms of the quality of the final solution for virtual population sizes of 30, 60, 100, 150 and 200. This suggests that the virtual population size has a minimal impact on the algorithm's performance within the specified range, indicating that the algorithm exhibits robust stability. Based on this outcome and in consideration of other compact literature, a smaller virtual population size (60) is recommended to effectively reduce the computational overhead while maintaining the algorithm's performance. The original GO algorithm has a population size of 30, while all compact algorithms have a virtual population size of 60. The maximum number of evaluations is 5000.

Table 1: Virtual population size analysis

Func_Num	Popsize: 30	Popsize: 60	Popsize: 100	Popsize: 150	Popsize: 200
1	13375216	13284980	13722765.1	14615128.3	14445327.6
2	4.37E+14	9.269E+14	1.6225E+15	2.6578E+14	1.3023E+15
3	36847.703	34763.877	34557.8458	35097.2726	36225.9131
4	481.11363	479.06303	480.435996	482.117192	473.829129
5	661.30318	660.19397	670.412389	649.383526	669.630762
6	650.24378	652.79724	647.870698	649.795044	653.054243
7	986.19698	995.48725	995.760155	993.665104	984.936236
8	939.49196	941.79452	935.513728	932.767998	936.689426
9	6846.9875	6400.6113	6140.98761	6171.07028	6305.13771
10	5186.0492	5253.1728	5087.14619	5114.84709	5022.58556
11	1213.623	1219.1138	1216.12012	1214.95404	1222.50445
12	5691885.3	5597414.9	6160738.94	4892217.51	5652533.93
13	322609.49	307782.41	322659.031	279250.383	284693.548
14	85846.596	78080.501	82289.8104	122817.091	92398.1928
15	13598.614	13568.856	14995.2279	13563.0811	12360.9083

(Continued)

Table 1 (continued)

Func_Num	Popsize: 30	Popsize: 60	Popsize: 100	Popsize: 150	Popsize: 200
16	2833.7027	2841.0092	2839.58244	2942.50385	3068.37128
17	2241.1074	2350.9087	2390.94695	2444.41457	2389.57237
18	382789.18	400331.72	351012.134	452969.536	350612.668
19	19674.304	24635.276	24664.0722	22800.1551	18001.8189
20	2606.7996	2619.1953	2598.00145	2667.08534	2626.95273
21	2462.0028	2455.7244	2462.95355	2447.30171	2464.0903
22	2323.8881	2324.9152	2325.46442	2537.88158	2504.22467
23	2922.8099	2914.3629	2914.77247	2958.13517	2953.96798
24	3014.772	3014.497	3011.69268	3000.78107	2999.80441
25	2895.8224	2891.2985	2898.45279	2898.31188	2893.99501
26	3682.2869	3734.2989	4105.03466	3634.93517	3472.2147
27	3215.8807	3213.985	3205.45635	3221.86241	3210.85475
28	3241.6258	3235.6123	3241.61051	3238.21408	3237.88099
29	4016.2269	4109.7012	4142.50136	4062.52159	4122.59491
30	369634.48	317089.72	361381.729	367580.346	461146.94

4.2 eGO Performance

Tables 2–4 show the means of the eight algorithms on 30, 50, and 100 dimensions of CEC2014, Tables 5–7 show the means of the eight algorithms on 30, 50, and 100 dimensions of CEC2017, respectively, and the metrics of each algorithm at a significance level of α of 0.05 and Wilcoxon signed rank test. Tables A1–A3 show the standard deviations and minimum values of the eight algorithms on 30, 50, and 100 dimensions of CEC2014. Tables A4–A6 show the standard deviations and minimum values of the eight algorithms on 30, 50, and 100 dimensions of CEC2017. All algorithms are presented in a single table, thus facilitating a clear comparison between them. The symbol '+' indicates that the eGO algorithm outperforms the other heuristics, while '=' indicates that the eGO algorithm performs as well as the other heuristics. The symbol '−' indicates that the eGO algorithm performs worse than the other heuristics. The summary row in each table shows that eGO outperforms the other algorithms. From the comparative results, it is evident that eGO has a significant advantage over the other compact algorithms and is highly competitive. It outperforms the other algorithms in most of the tested functions. In the comparison between eGO and GO, eGO obtain more 'winning values' in different dimensions. However, it is less effective than GO in optimizing certain functions.

Table 2: Comparison of eGO, eGO_l, GO, and commonly used compact algorithms on 30 dimensions of CEC2014 and the Wilcoxon signed rank test

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
1	2.12E+09	+	4.76E+08	+	1.73E+09	+	3.142E+09	+	809405472	+	20382384	+	30142318.1	+	9110778.9
2	1.08E+11	+	3.57E+10	+	9.677E+10	+	1.067E+11	+	4.64E+10	+	1.01E+08	+	325234702	+	16590521
3	271084	+	81802.33	+	211525.98	+	143048917	+	131117.82	+	8863.416	−	15664.6457	+	12540.132
4	24890.84	+	4717.186	+	21377.934	+	28070.067	+	4878.092	+	550.4643	+	544.103395	+	485.61213
5	521.1977	+	520.4831	−	520.93179	−	521.68869	+	521.11098	+	521.0727	=	521.107655	+	521.06038
6	643.2464	+	649.1186	+	644.91286	+	652.39351	+	640.85764	+	626.1486	+	625.044935	+	622.59485
7	1671.762	+	1078.39	+	1524.432	+	1835.1541	+	1114.4384	+	701.9849	+	708.552484	+	701.22893

(Continued)

Table 2 (continued)

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
8	1266.089	+	967.4823	+	1201.138	+	1264.0515	+	1130.4413	+	908.888	=	1028.89842	+	910.76135
9	1456.553	+	1088.361	+	1346.5968	+	1399.0178	+	1250.6696	+	1103.658	+	1153.41014	+	1051.7419
10	9692.385	+	5389.351	+	9406.5047	+	12049.42	+	8462.5633	+	5202.77	+	5877.92974	+	3148.6411
11	9344.345	+	6231.106	+	9695.8712	+	13079.015	+	9212.1948	+	8280.311	+	7547.67467	+	4799.2276
12	1204.099	+	1202.618	+	1204.093	+	1214.1029	+	1203.6228	+	1203.023	+	1203.29429	+	1201.6898
13	1309.329	+	1305.837	+	1308.8045	+	1311.0428	+	1305.2622	+	1300.599	–	1300.8901	+	1300.7287
14	1723.55	+	1557.213	+	1689.4082	+	1827.6951	+	1538.5195	+	1400.419	=	1401.59186	+	1400.4026
15	10290336	+	3729.784	+	939190.44	+	3212143.8	+	238248.5	+	1525.714	=	1996.59278	+	1529.4314
16	1613.7	+	1614.446	+	1613.8612	+	1614.9984	+	1613.4203	+	1613.218	+	1612.87483	+	1612.6431
17	1.54E+08	+	40346038	+	104223398	+	1.125E+09	+	37892627	+	482917.8	–	5803028.33	+	689685.41
18	5.05E+09	+	6.88E+08	+	3.986E+09	+	1.543E+10	+	1.09E+09	+	52377.39	–	1058872.29	+	76722.725
19	2678.838	+	2147.637	+	2468.7116	+	3062.9857	+	2086.6538	+	1916.687	+	1928.85676	+	1915.5559
20	2857821	+	152998.4	+	1465866.3	+	4.055E+09	+	238142.25	+	5597.085	–	25630.7743	+	19787.582
21	71578885	+	16763622	+	47625970	+	3.075E+09	+	11848428	+	98419.84	–	2600688.96	+	219014.26
22	4542.836	+	5127.601	+	4662.8066	+	6907807	+	3673.0781	+	2926.227	=	2910.87533	=	2869.9888
23	3715.487	+	2512.266	–	2503.302	–	3663.9747	+	2817.1	+	2617.307	+	2580.30126	=	2582.3884
24	2880.637	+	2604.483	–	2600.9961	–	2717.9047	+	2657.8827	+	2639.92	+	2610.24178	+	2608.0011
25	2851.474	+	2700.196	–	2700.0479	–	2749.933	+	2730.1293	+	2711.383	+	2701.39295	=	2701.349
26	2764.392	=	2800.006	+	2719.0916	–	2832.3865	+	2715.8413	–	2700.616	–	2795.1849	+	2795.0825
27	4276.82	+	2912.687	–	2901.379	–	23824.423	+	3985.1498	+	3353.512	=	3243.78931	=	3315.3641
28	9364.533	+	3017.553	–	3001.9965	–	20366.138	+	6479.1736	+	3904.67	+	3524.40793	=	3458.1946
29	2.69E+08	+	14446804	+	3899563.2	–	1.594E+09	+	60276529	+	454835.5	=	15919628.9	+	11878378
30	4105591	+	996006.9	+	255236.22	+	115431888	+	1044629.1	+	10167.77	–	936836.553	+	221422.34
win		29		24		22		30		29		19		28	

Table 3: Comparison of eGO, eGO_l, GO, and commonly used compact algorithms on 50 dimensions of CEC2014 and the Wilcoxon signed rank test

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
1	5.81E+09	+	2.59E+09	+	4.999E+09	+	1.719E+10	+	2.014E+09	+	96384710	+	107031956	+	19159419
2	2.53E+11	+	9.17E+10	+	1.967E+11	+	2.116E+11	+	1.072E+11	+	2.15E+09	+	1880177307	+	86359453
3	340640.3	+	144878.1	+	375400.01	+	699765347	+	251407.67	+	78158.2	+	47625.4751	+	37839.065
4	86733.18	+	19519.76	+	65804.407	+	79922.84	+	25009.668	+	949.7479	+	1001.57055	+	577.84413
5	521.333	+	520.7578	–	521.1156	–	521.72909	+	521.27014	+	521.2466	=	521.251042	=	521.2391
6	680.8614	+	685.497	+	680.00575	+	689.06549	+	675.46558	+	653.0019	+	648.202777	=	645.20586
7	2975.248	+	1557.005	+	2473.966	+	2661.3267	+	1749.0962	+	722.6054	+	725.344428	+	702.26357
8	1689.053	+	1103.199	+	1530.0393	+	1621.0168	+	1442.449	+	1101.773	+	1296.08566	+	1063.1687
9	2027.977	+	1335.633	+	1832.2102	+	1850.1221	+	1602.5454	+	1339.888	+	1397.56567	+	1247.6163
10	16568.29	+	9335.264	+	16179.816	+	19999.114	+	15215.823	+	11184.78	+	12569.9723	+	6201.5879
11	16732.36	+	10275.19	+	16541.753	+	21033.161	+	16003.048	+	15104.54	+	14297.5821	+	9478.5907
12	1204.661	+	1204.994	+	1205.3327	+	1213.1372	+	1205.125	+	1204.228	+	1204.49514	+	1202.3955
13	1311.378	+	1306.196	+	1309.6273	+	1310.1714	+	1307.0432	+	1300.734	=	1301.03235	+	1300.7081
14	1994.82	+	1609.173	+	1783.7212	+	1892.6886	+	1680.9179	+	1401.398	+	1417.21168	+	1400.4232
15	79620359	+	246641.6	+	23825811	+	33133794	+	2126405.9	+	1940.517	+	23398.5515	+	1562.9132
16	1623.782	+	1623.755	+	1623.1623	+	1625.0021	+	1623.2026	+	1623.094	+	1622.58179	+	1622.2929
17	7.65E+08	+	3.58E+08	+	578934251	+	3.927E+09	+	214579161	+	3828782	+	20734870.6	+	2249784.8
18	2.16E+10	+	9.03E+09	+	1.834E+10	+	3.868E+10	+	4.714E+09	+	158404	–	15498816.9	+	1045176
19	5127.233	+	2734.029	+	4536.4035	+	11438.716	+	2637.8798	+	1961.446	+	1947.30332	+	1926.9161
20	6158762	+	104794.2	+	2711140.6	+	3.654E+09	+	482092.63	+	23841.2	–	30539.5352	=	27669.636
21	2.38E+08	+	10434448	+	160129845	+	1.971E+09	+	52511763	+	1210162	=	5238860.27	+	1428677.2
22	113190.2	+	30168.06	+	54398.948	+	7441063.3	+	6214.3457	+	4124.638	+	4325.13793	+	3669.6363
23	5167.8	+	2525.084	–	2506.4299	–	4376.7658	+	3935.672	+	2658.946	+	2619.18211	=	2620.0131
24	3230.404	+	2611.334	–	2602.4823	–	2838.82	+	2823.0319	+	2700.267	+	2616.71509	=	2616.1617
25	3122.149	+	2700.556	–	2700.1422	–	2797.0117	+	2771.9833	+	2738.955	+	2702.75021	=	2702.8223
26	3127.752	+	2800.023	+	2793.158	–	2862.6572	+	2833.7873	+	2748.906	=	2800.25852	+	2795.1911
27	5426.354	+	2941.414	–	2903.0256	–	39980.7	+	5039.0205	+	4148.082	+	3712.56764	=	3726.3827
28	15835.3	+	3077.629	–	3012.3017	–	35161.105	+	11950.96	+	5017.184	+	3929.0501	=	3913.9669
29	1.71E+09	+	47157899	+	11728102	+	3.765E+09	+	491548572	+	6627744	–	5397049.93	–	9253525.8
30	41207893	+	2041872	+	522737.79	+	148756664	+	7921996.3	+	49594.64	–	1141325.12	+	453976.67
win		30		24		23		30		30		24		26	

Table 4: Comparison of eGO, eGO_l, GO, and commonly used compact algorithms on 100 dimensions of CEC2014 and the Wilcoxon signed rank test

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
1	1.75E+10	+	3.4E+09	+	1.54E+10	+	1.904E+10	+	6.835E+09	+	6.14E+08	+	338167820	+	125694611
2	5.85E+11	+	1.97E+11	+	3.382E+11	+	3.552E+11	+	2.695E+11	+	3.55E+10	+	9733845381	+	704776687
3	803742.2	+	297705	+	770873.64	+	330486218	+	513193.01	+	274814.4	+	175202.048	+	137229.13
4	234964.4	+	46455.87	+	139000.64	+	151947.9	+	69617.205	+	3867.689	+	2193.33855	+	899.66232
5	521.4232	+	521.0764	–	521.31289	–	521.70186	+	521.41598	=	521.4181	=	521.411338	=	521.40915
6	767.9049	+	772.9743	+	769.48079	+	781.19858	+	762.68696	+	736.0323	+	723.493563	+	718.29575
7	5989.598	+	2767.325	+	4135.5676	+	4268.7601	+	3329.6932	+	1013.111	+	804.353201	+	708.05749
8	2754.736	+	1409.892	–	2296.613	+	2441.0292	+	2263.0448	+	1661.243	+	2012.89235	+	1526.489
9	3261.64	+	1701.426	–	2552.3049	+	2751.3572	+	2486.2976	+	2033.043	+	2147.01393	+	1880.2212
10	34685.11	+	18361.28	+	34232.004	+	39117.509	+	32690.647	+	26881.2	+	29939.7485	+	17589.286
11	34989.49	+	15698.86	–	34285.159	+	39902.63	+	34133.791	+	33331.43	+	31237.07	+	21799.508
12	1205.701	+	1204.268	+	1205.7099	+	1211.0183	+	1205.1945	+	1204.971	+	1205.08179	+	1203.6127
13	1313.359	+	1307.571	+	1310.1392	+	1310.4478	+	1308.8748	+	1302.6	+	1302.03445	+	1300.8116
14	2974.291	+	2011.298	+	2430.1153	+	2483.4246	+	2172.8158	+	1494.943	+	1449.88712	+	1400.7501
15	3.73E+08	+	2559117	+	56838592	+	80336192	+	16462235	+	38868.46	+	185996.699	+	1676.7234
16	1648.249	+	1647.92	+	1647.8006	+	1649.8235	+	1647.8222	+	1647.723	+	1646.97543	=	1646.8343
17	1.86E+09	+	6.75E+08	+	2.206E+09	+	3.856E+09	+	807365794	+	53501214	+	49137777.8	+	11827774
18	6.55E+10	+	2.05E+10	+	5.621E+10	+	6.239E+10	+	2.142E+10	+	35624404	+	90970092	+	8785019.2
19	16587.86	+	5481.202	+	14857.436	+	20129.137	+	5582.9134	+	2154.572	+	2111.68953	+	2011.4667
20	20640493	+	556063.7	+	15415483	+	1.415E+09	+	2157144.5	+	212686.3	+	161820.851	+	104769.92
21	1.16E+09	+	1.14E+08	+	1.042E+09	+	1.753E+09	+	390041208	+	18263169	+	15558558.5	+	5442079.2
22	758476.6	+	16395.39	+	535392.72	+	3012745.7	+	12295.863	+	7241.61	+	5651.10719	+	5237.388
23	9408.839	+	2537.291	–	2507.7548	–	4127.952	+	4040.7572	+	2665.325	+	2606.8199	=	2607.9457
24	2656.564	+	2609.915	+	2602.0674	+	3151.0411	+	2596.3845	+	2500.562	+	2500.92888	+	2500.2059
25	2683.7	+	2698.492	+	2682.5361	+	2695.5133	+	2669.146	+	2614.062	+	2612.84185	+	2603.7041
26	2660.608	+	2771.683	+	2765.0619	+	2800.0024	+	2622.483	+	2625	+	2610	+	2600
27	3000.515	+	2900.084	–	2900.0178	–	2905.8514	+	2905.7382	+	3000	+	2900.25131	=	2900.2494
28	3097.501	+	3000.714	–	3000.1512	–	3030.444	+	3029.6952	+	3027.328	+	3002.10289	=	3001.8864
29	3080.962	+	3097.754	+	3078.8326	+	3092.9704	+	3064.2372	+	3016.415	+	3016.55418	+	3004.5862
30	3183.689	+	3191.76	+	3181.3283	+	3195.2091	+	3168.2656	+	3114.468	+	3113.1019	+	3103.9162
win		30		23		26		30		30		30		29	

Table 5: Comparison of eGO, eGO_l, GO, and commonly used compact algorithms on 30 dimensions of CEC2017 and the Wilcoxon signed rank test

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
1	9.97E+10	+	2.48E+10	+	8.01E+10	+	8.79E+10	+	3.23E+10	+	8.05E+07	+	2.25E+08	+	13284980
2	1.89E+45	+	3.64E+48	+	3.90E+45	+	1.25E+62	+	4.10E+41	+	1.71E+28	+	1.90E+31	+	9.27E+14
3	369906.6	+	67132.8	+	209310	+	3.24E+11	+	163562.4	+	60523.53	+	68468.31	+	34763.88
4	30552.93	+	5984.945	+	22312.03	+	38085.02	+	5960.438	+	538.2364	+	529.4128	+	479.063
5	1080.801	+	804.085	+	1030.845	+	1098.92	+	881.1124	+	702.7123	+	746.6693	+	660.194
6	717.653	+	667.2539	+	707.8109	+	744.0642	+	682.0942	+	608.3748	–	633.993	–	652.7972
7	2938.051	+	1346.867	+	1566.434	+	1721.777	+	1405.711	+	967.1037	–	1010.782	+	995.4872
8	1328.579	+	1000.236	+	1237.901	+	1315.359	+	1144.399	+	1005.84	+	1038.231	+	941.7945
9	27235.3	+	5882.47	–	19553.46	+	29737.92	+	12751.66	+	1917.723	–	5647.795	–	6400.611
10	9783.834	+	5494.64	+	9526.76	+	12618.53	+	9142.749	+	8259.118	+	7612.424	+	5253.173
11	22844.89	+	3269.829	+	19909.71	+	7.54E+08	+	7248.634	+	1366.862	+	2884.233	+	1219.114
12	1.69E+10	+	6.27E+09	+	1.38E+10	+	3.06E+10	+	4.61E+09	+	4145670	–	30828494	+	5597415
13	1.13E+10	+	4.97E+09	+	8.96E+09	+	4.59E+10	+	2.86E+09	+	144008.7	–	9223501	+	307782.4
14	8869610	+	1152564	+	7901506	+	1.22E+09	+	2148279	+	10206.95	–	1025965	+	78080.5
15	2.41E+09	+	17608.08	+	1.79E+09	+	8.09E+09	+	1.02E+08	+	81117.13	+	41647.19	+	13568.86
16	5858.209	+	6001.191	+	6016.106	+	25906.3	+	4491.143	+	3303.154	+	2950.4	=	2841.009
17	4927.459	+	3105.618	+	4262.457	+	400822.8	+	3183.679	+	2312.41	=	2379.568	=	2350.909
18	1.47E+08	+	4451039	+	1.07E+08	+	6.08E+09	+	35766409	+	421994.7	=	2600313	+	400331.7
19	3.23E+09	+	1290297	+	2.19E+09	+	8.31E+09	+	2.34E+08	+	53197.39	+	79409.69	+	24635.28
20	3471.888	+	3295.518	+	3319.77	+	4732.431	+	3102.922	+	2723.564	+	2660.158	=	2619.195
21	2809.592	+	2729.577	+	2811.565	+	3251.171	+	2658.301	+	2501.218	+	2551.885	+	2455.724
22	10875.65	+	8192.652	+	10642.76	+	12934.4	+	8949.913	+	5107.68	+	2340.714	+	2324.915

(Continued)

Table 5 (continued)

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
23	3305.875	+	4276.047	+	3565.578	+	7611.941	+	3135.965	+	2861.218	–	2926.73	=	2914.363
24	3508.64	+	4270.965	+	3784.062	+	5290.326	+	3290.529	+	3046.489	+	3054.301	+	3014.497
25	13019.77	+	3421.908	+	7307.727	+	10223.61	+	4533.311	+	2920.821	+	2961.853	+	2891.298
26	10551.65	+	9545.482	+	12398.09	+	16577.63	+	8583.228	+	5851.38	+	3616.997	–	3734.299
27	4448.791	+	6895.028	+	4445.418	+	10571.7	+	3696.293	+	3231.084	+	3280.343	+	3213.985
28	10264.79	+	5048.604	+	9223.019	+	11031.61	+	5164.535	+	3302.208	+	3308.977	+	3235.612
29	6910.009	+	6800.881	+	7688.808	+	496593.3	+	5630.029	+	4158.746	=	4287.285	+	4109.701
30	1.87E+09	+	3.56E+08	+	1.32E+09	+	1.18E+10	+	4.09E+08	+	279219.2	=	1751147	+	317089.7
win		30		29		30		30		30		21		27	

Table 6: Comparison of eGO, eGO_l, GO, and commonly used compact algorithms on 50 dimensions of CEC2017 and the Wilcoxon signed rank test

Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
1	2.17E+11	+	5.90E+10	+	1.34E+11	+	1.43E+11	+	8.75E+10	+	2.14E+09	+	1.30E+09	+	70933702
2	8.35E+85	+	2.68E+68	+	6.40E+83	+	1.60E+93	+	1.64E+78	+	4.43E+56	+	1.28E+59	+	1.36E+34
3	11578178	+	164996.1	+	477175	+	5.08E+14	+	317011.8	+	187005.2	+	159258.7	+	124403.4
4	78739.17	+	15543.98	+	55943.03	+	60866.22	+	22681.36	+	915.7304	+	845.1439	+	596.878
5	1581.881	+	880.179	+	1319.914	+	1423.144	+	1210.746	+	940.3394	+	1016.304	+	838.2563
6	734.1185	+	671.146	+	708.7597	+	746.8121	+	702.6068	+	622.2573	–	656.3782	–	665.3134
7	5411.435	+	1798.305	+	2107.397	+	2497.125	+	2165.569	+	1288.665	–	1342.143	=	1344.334
8	1865.971	+	1220.559	+	1679.401	+	1754.995	+	1502.83	+	1240.846	+	1300.698	+	1144.248
9	80218.15	+	16189.64	–	47015.83	+	73517.17	+	44573.92	+	9919.749	–	25476.25	=	25778.67
10	16449.28	+	10107.32	+	16271.57	+	20799.61	+	15935.65	+	14861.51	+	13782.76	+	8671.168
11	54636.92	+	10163.76	+	53061.88	+	4.51E+08	+	19421.11	+	2901.955	+	7815.683	+	1479.189
12	1.00E+11	+	4.77E+10	+	8.68E+10	+	1.49E+11	+	3.58E+10	+	1.29E+08	+	3.73E+08	+	38178371
13	5.03E+10	+	2.75E+10	+	4.14E+10	+	1.16E+11	+	1.22E+10	+	881137.5	–	65714064	+	2143052
14	69814369	+	75697919	+	73342176	+	1.63E+09	+	20782769	+	138036.6	–	4561897	+	411684.1
15	1.81E+10	+	6.46E+08	+	1.55E+10	+	2.59E+10	+	2.08E+09	+	158588.2	–	3312979	+	324064.1
16	10316.83	+	6764.248	+	10350.17	+	23071.01	+	6694.749	+	5078.377	+	3519.002	=	3519.995
17	158349.9	+	3710.362	+	67956.05	+	466707.3	+	5843.854	+	3967.829	+	3321.567	=	3255.556
18	5.20E+08	+	32730806	+	3.37E+08	+	2.49E+09	+	89963320	+	1462638	–	9507034	+	2014925
19	6.59E+09	+	4.07E+08	+	5.39E+09	+	1.53E+10	+	1.35E+09	+	98771.81	–	1641377	+	170319.9
20	4953.548	+	3850.932	+	4823.51	+	6738.623	+	4439.783	+	3977.766	+	3639.117	+	3216.857
21	3360.182	+	3000.045	+	3333.811	+	4348.766	+	3085.965	+	2727.612	+	2815.052	+	2669.225
22	17661.58	+	13891.58	+	18186.17	+	21801.54	+	17727.93	+	16464.53	+	5568.338	=	9221.982
23	4085.251	+	4947.42	+	4530.965	+	9619.929	+	3803.41	+	3193.826	–	3349.891	=	3380.162
24	4425.347	+	5673.19	+	4976.278	+	6887.61	+	4024.335	+	3380.012	+	3331.023	+	3245.783
25	43252.52	+	8164.381	+	19558	+	21389.79	+	12803.09	+	3333.419	+	3208.36	+	3062.245
26	19801.56	+	13621.9	+	19995.99	+	22094.37	+	15647.2	+	8537.774	+	4596.753	+	3414.54
27	7317.854	+	12289.44	+	6989.589	+	18689.91	+	5264.981	+	3479.178	=	3779.142	+	3505.812
28	19060.29	+	8822.933	+	18109.66	+	21295.24	+	10287.9	+	3785.307	+	3634.967	+	3311.789
29	149543.6	+	24536.42	+	81677.68	+	9508528	+	12667.21	+	5531.174	+	5796.934	+	5034.02
30	1.05E+10	+	2.09E+09	+	8.62E+09	+	2.49E+10	+	2.99E+09	+	19163938	+	41707882	+	11848344
win		30		29		30		30		30		20		24	

Table 7: Comparison of eGO, eGO_l, GO, and commonly used compact algorithms on 100 dimensions of CEC2017 and the Wilcoxon signed rank test

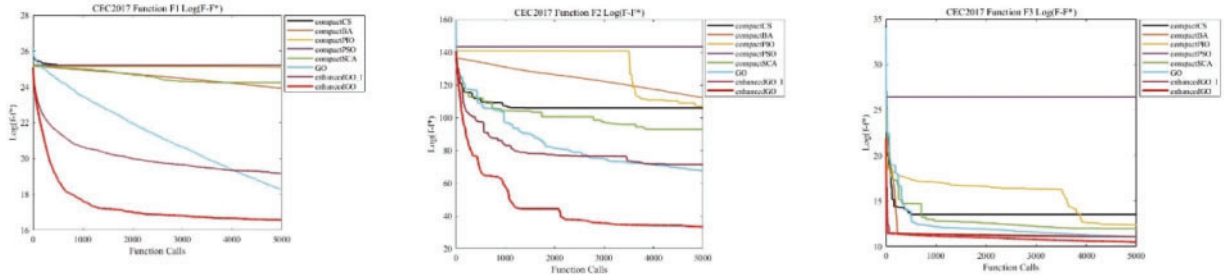
Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_l	R	Mean_eGO
1	5.44E+11	+	1.71E+11	+	2.95E+11	+	3.19E+11	+	2.47E+11	+	3.18E+10	+	9.68E+09	+	6.89E+08
2	4.21E+185	+	8.45E+152	+	2.09E+182	+	1.20E+202	+	1.61E+171	+	1.24E+138	+	2.55E+145	+	5.46E+105
3	4.67E+08	+	342517.9	+	1023963	+	1.48E+15	+	942879.7	+	513415.1	+	360364.6	+	335506.8
4	228729.4	+	51017.01	+	156456.2	+	169022.5	+	71555.91	+	4310.699	+	2448.633	+	954.0446
5	2938.836	+	1385.773	–	2323.606	+	2429.356	+	2183.207	+	1653.123	+	1766.046	+	1506.722

(Continued)

Table 7 (continued)

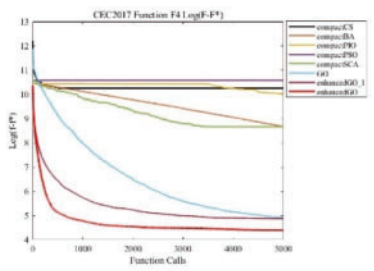
Func_Num	Mean_cCS	R	Mean_cBA	R	MeancPIO	R	MeancPSO	R	MeancSCA	R	Mean_GO	R	Mean_eGO_1	R	Mean_eGO
6	751.9415	+	670.8426	–	721.3684	+	741.5117	+	712.3145	+	653.1237	–	680.1438	=	681.1343
7	12168.65	+	3376.292	+	4183.285	+	4695.162	+	4546.182	+	2384.097	–	2371.284	–	2516.952
8	3339.97	+	1897.92	+	2783.885	+	2943.972	+	2535.533	+	1955.548	+	2074.539	+	1856.845
9	193603.5	+	33560.13	–	77451.51	+	134883.6	+	107951.9	+	50510.35	–	66867.55	=	67359.71
10	34887.52	+	17910.1	–	34363.59	+	39816.64	+	33779.84	+	32896.03	+	30858.98	+	21504.65
11	629912.6	+	151704.9	+	512338.3	+	3.01E+13	+	245314.5	+	110118.8	+	89207.34	+	34812.74
12	3.02E+11	+	1.22E+11	+	2.52E+11	+	2.77E+11	+	1.34E+11	+	3.78E+09	+	2.72E+09	+	4.24E+08
13	7.47E+10	+	2.50E+10	+	6.10E+10	+	7.04E+10	+	2.51E+10	+	48900947	+	1.86E+08	+	9386286
14	3.15E+08	+	9606898	+	3.21E+08	+	1.70E+09	+	92934285	+	3963053	=	7433108	+	3000556
15	3.56E+10	+	1.10E+10	+	2.94E+10	+	4.48E+10	+	9.13E+09	+	1597624	–	23146555	+	2200469
16	31473.82	+	14870.45	+	27392.61	+	38989.09	+	17036.69	+	11067.18	+	7618.127	+	6777.017
17	23893702	+	829988.9	+	10277722	+	2.40E+08	+	203053.5	+	7767.361	+	18972.55	+	5375.514
18	5.82E+08	+	8299208	+	5.65E+08	+	1.49E+09	+	2.01E+08	+	6978216	+	6279867	+	3289257
19	3.45E+10	+	1.12E+10	+	3.17E+10	+	4.30E+10	+	7.95E+09	+	1892615	–	23635722	+	2274505
20	8472.158	+	6512.168	+	8853.773	+	11114.46	+	8410.681	+	8050.305	+	7128.667	+	5477.43
21	4999.666	+	8252.683	+	5037.588	+	10968.66	+	4361.507	+	3482.567	=	3556.635	+	3472.343
22	36807.33	+	23250.91	–	36927.57	+	40868.3	+	35776.99	+	34956.2	+	19616.6	=	25704.19
23	7211.217	+	7267.146	+	6918.447	+	16622.7	+	5491.698	+	3971.905	=	4166.649	+	3999.292
24	11864.75	+	11751.02	+	11518.72	+	16985.06	+	7903.814	+	4592.002	=	4779.854	+	4568.52
25	110665.2	+	15351.87	+	35224.97	+	38908.98	+	28516.07	+	6399.433	+	3857.82	+	3515.652
26	61892.43	+	39566.84	+	65550.79	+	71241.57	+	44567.51	+	18907.63	+	16020.88	+	11264.5
27	14312.85	+	17842.09	+	13562.22	+	25543.75	+	9407.435	+	4055.698	+	3930.803	+	3670.076
28	62813.13	+	23360.45	+	43038.94	+	46030.77	+	29477.41	+	8562.547	+	3961.058	+	3509.701
29	4043958	+	71710.36	+	2155873	+	11770414	+	76948.37	+	10773.71	+	13445.95	+	8331.113
30	5.37E+10	+	2.43E+10	+	4.88E+10	+	6.16E+10	+	1.96E+10	+	34608877	+	2.39E+08	+	26440632
win		30		25		30		30		30		24		26	

To determine the effectiveness of the algorithm, it is essential to analyze both the optimal value achieved by the algorithm and the convergence of the algorithm. To better illustrate the data changes, we calculated the difference between the average value of the algorithm and the optimal value given by CEC for log. In this paper, a CEC2017-30-dimensional convergence graph is chosen as a representative for convergence analysis. The performance of the eGO algorithm proposed in this paper is compared with that of other algorithms in terms of convergence in Fig. 2. The horizontal axis represents the number of evaluations, while the vertical axis represents the logarithm of the difference between the function value f and the optimal value f^* ($log\,(f - f^*)$). A lower value on the vertical axis indicates better convergence, as it signifies a smaller difference between f and f^*. The convergence curves indicate that the eGO algorithm has a stronger convergence ability than the other algorithms in general.

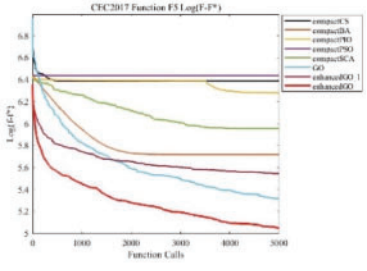


F1 F2 F3

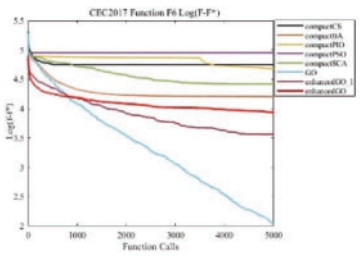
Figure 2: (Continued)



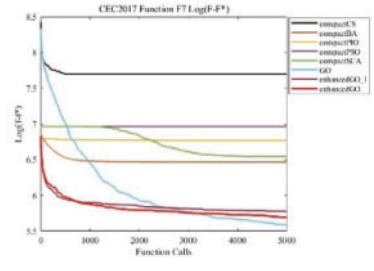
F4



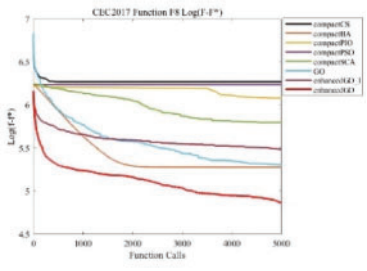
F5



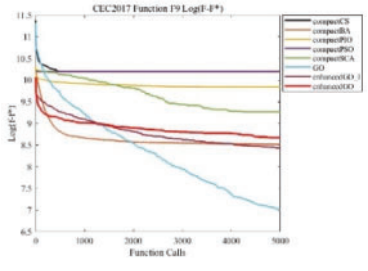
F6



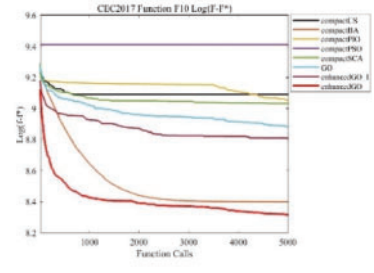
F7



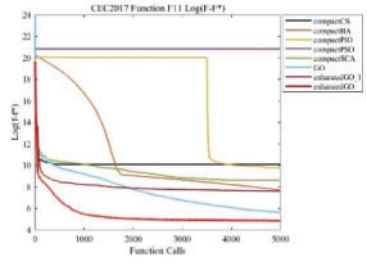
F8



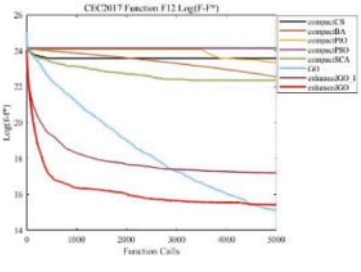
F9



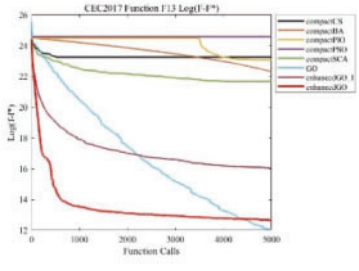
F10



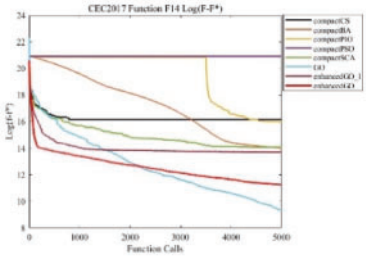
F11



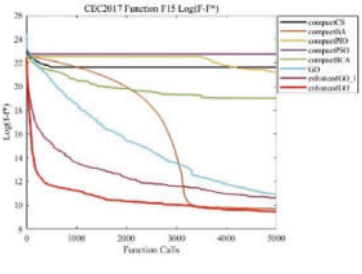
F12



F13

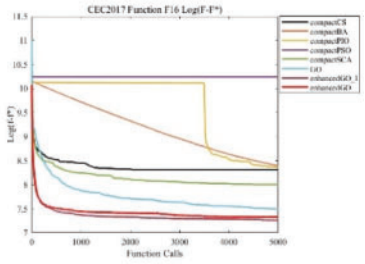


F14

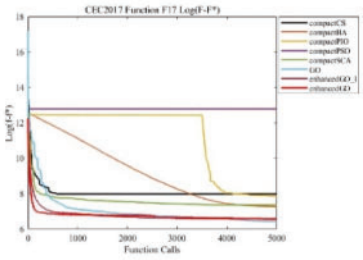


F15

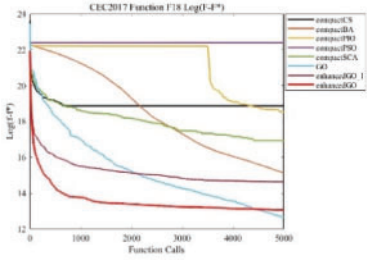
Figure 2: (Continued)



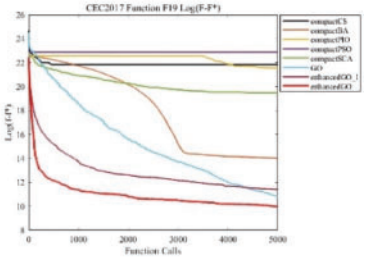
F16



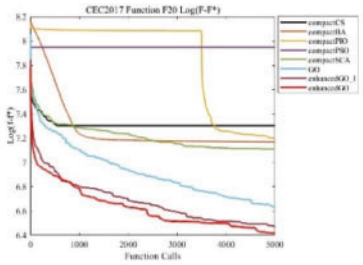
F17



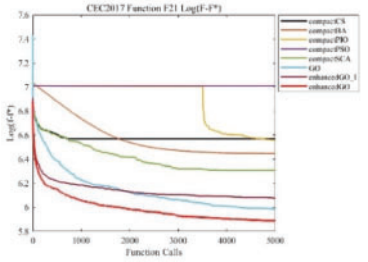
F18



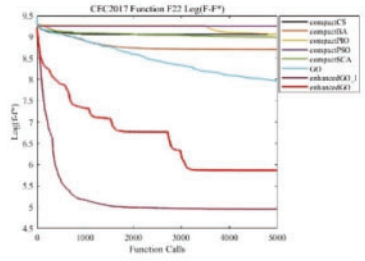
F19



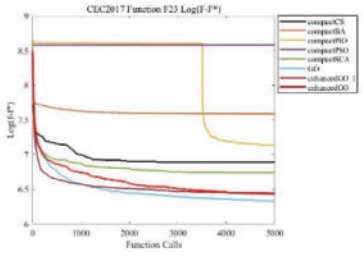
F20



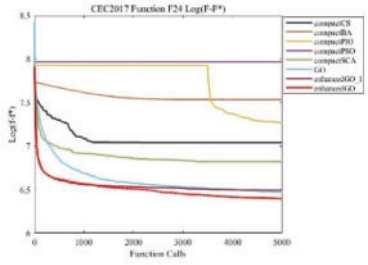
F21



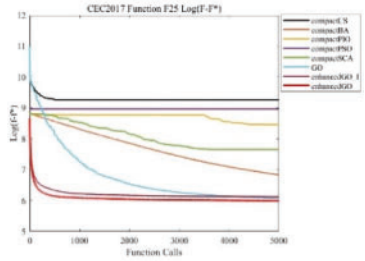
F22



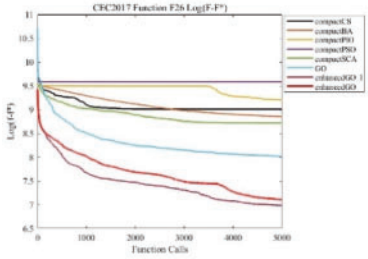
F23



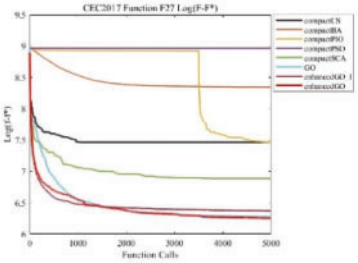
F24



F25

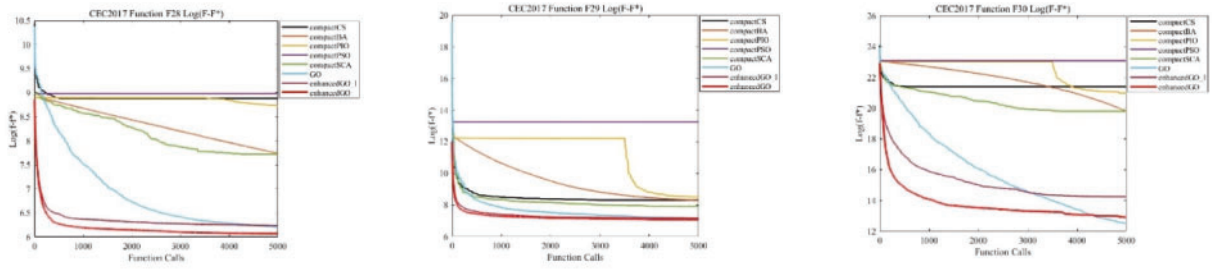


F26



F27

Figure 2: (Continued)



F28 F29 F30

Figure 2: F1–F30 convergence graph of CEC2017 benchmark function on 30-D

5 Case of Study: Enhanced Growth Optimizer for Image Fusion

This section implements image fusion using the eGO algorithm and compares it to traditional methods. Image fusion uses single-band panchromatic (*PAN*) images and multi-band multispectral (*MS*) images. The process of merging these two types of images to produce images with enhanced spectral and spatial resolution is commonly known as pan-sharpening. We use the Eq. (17).

$$MS_k^H = MS_k^L + g_k \times D \tag{17}$$

where MS^H represents the high-resolution multispectral band, MS^L represents the low-resolution multispectral image, g_k is a real number representing the mixing parameter, k represents the kth wave spectrum of the multispectral image, and D represents the spatial detail map in Eq. (18).

$$D = P - I \tag{18}$$

where P is the panchromatic image and I is the intensity component in the IHS, calculated as follows Eq. (19).

$$I = \sum_{i=1}^{n} \alpha_i MS_k^L \tag{19}$$

where α_i represents a non-negative coefficient less than 1, and it has an average value of 1/3 in RGB images; however, multispectral images have four bands, with RGB and an infrared band, so α_i is often defined as $1/n$ for multispectral images, where n represents the number of channels. To minimize spectral distortion in IHS panchromatic sharpened images, the calculation of intensity bands is modified based on the original multispectral and panchromatic images. To minimize spectral distortion, the intensity bands should be as close as possible to the panchromatic image. Therefore, in this adaptive IHS method, we want to find the closest coefficient α_i that satisfies the $P \approx \sum_{i=1}^{n} \alpha_i M_i$ [23,24]. Hence, it is necessary to solve the optimization problem stated in Eq. (20) in order to reduce the impact of low-frequency components.

$$\|D\|^2 = \|P - \sum_{i=1}^{3} \alpha_i MS_i^L\|^2 \tag{20}$$

The optimal value of α_i can be calculated using the Least Squares Method (LSM) as follows Eq. (21).

$$\alpha_i = \left(\left(MS_i^L\right)^T MS_i^L\right)^{-1} \left(MS_i^L\right)^T P. \tag{21}$$

The seven algorithms described above are used to adaptively tune the α_i. For ease of understanding, Eq. (17) is modified as follows Eq. (22).

$$MS_k^H = MS_k^L + g_k \times \Gamma_k \odot D \tag{22}$$

where Γ_k is a weighting matrix defined as follows Eqs. (23) and (24).

$$\Gamma_k = \mathrm{B}(k)\,\Gamma_p + (1 - \mathrm{B}(k))\,\Gamma_{MS_k} \tag{23}$$

$$\Gamma_p = \exp\left(-\frac{\lambda}{|\nabla P|^4 + \epsilon}\right), \Gamma_{MS_k} = \exp\left(-\frac{\lambda}{|\nabla MS_k|^4 + \epsilon}\right) \tag{24}$$

The gradient of the panchromatic map (∇P) and the gradient of each spectral channel of the multispectral map (∇MS) are used in this equation, along with a parameter (λ) to control image smoothness. Additionally, a very small value (ϵ) is included to prevent division by zero in the denominator. This paper uses the relative dimensionless global error in synthesis (ERGAS) between hybrid and low-resolution multispectral images to compute the parameter β for adaptive Pan-Sharpening. The formula for ERGAS is Eq. (25).

$$ERGAS = 100\frac{h}{l}\sqrt{\frac{1}{N}\sum_{i=1}^{n}\left(\frac{RMSE(n)}{\mu(n)}\right)^2} \tag{25}$$

where h represents the resolution of the high resolution image, l represents the resolution of the low resolution image, μ is the mean value of the nth band, and N is the number of bands. $RMSE$ is the root mean square error, the formula for calculating $RMSE$ is as follows Eq. (26).

$$RMSE = \sqrt{\frac{1}{L \times K}\sum_{i=1}^{L}\sum_{i=1}^{K}(F(i,j) - MS(i,j))^2} \tag{26}$$

The high-resolution hybrid image is represented by F, while the original low-resolution multispectral image is represented by MS. The size of the image is denoted by $L \times K$. Experiments were conducted on a dataset collected by QuickBird, which includes a high-resolution Pan image and a low-resolution MS image composed of three different bands (red, green, and blue). The proposed algorithm's performance is compared to six other algorithms mentioned above, as well as pso, using the $ERGAS$ index as an evaluation function and calculating the coefficients in Eq. (21). In order to do a quantitative assessment of various picture fusion techniques, we employ five overarching metrics. These include the relative average spectral error ($RASE$) [25], correlation coefficient (CC) [26], universal image quality index ($UIQI$) [27], and spectral angle mapper (SAM). $RASE$ represents the average performance of the spectral band, with lower values indicating higher spectral quality. Its calculation formula is as follows Eq. (27).

$$RASE = \frac{100}{M}\sqrt{\frac{1}{N}\sum_{i=1}^{N}RASE^2(n)} \tag{27}$$

CC is utilized to compare the degree of spectral distortion between the original multispectral bands and the final fused image bands. A higher CC value indicates better pan-sharpening and

less distortion, with a maximum value of 1. *UIQI* quantifies the extent to which there is a lack of correlation, distortion in brightness, and distortion in contrast. A higher *UIQI* value indicates better maintained spectral quality, with a maximum value of 1. The *SAM* metric quantifies the magnitude of the spectral angle between the sharpened result and the reference vector. A smaller *SAM* indicates a better sharpened result, with a minimum value of zero.

The fused image was created by applying Eq. (22) to the refined detail maps. The evaluation results of the image fusion are presented in Table 8. Fig. 3a,b shows the original *MS* and *PAN* maps, and the other images in Fig. 3 show the images after fusion by each method. It is evident from the graph that the images fused by the eGO algorithm produce better results than those produced by other algorithms.

Table 8: Evaluation results

Row	ERGAS	SAM	RASE	RMSE	UIQI	CC
cBA	5.244518885	6.396928558	19.9250643	19.39013686	0.93418118	0.922067
cPIO	5.214316138	6.35969727	19.84013579	19.30748842	0.934783827	0.923119
cPSO	5.232815529	6.38426651	19.89745918	19.36327285	0.934362704	0.922444
cSCA	5.231936042	6.382634647	19.89386824	19.35977832	0.934393119	0.922451
GO	5.229654739	6.378432531	19.88383429	19.35001374	0.934462444	0.922556
PSO	5.222671967	6.365251	19.86227252	19.32903084	0.934653179	0.922809
eGO	5.17701682	6.197201479	19.75714395	19.22672465	0.936061847	0.924845

(a) MS **(b) PAN** **(c) cBA**

(d) cPIO **(e) cPSO** **(f) cSCA**

Figure 3: (Continued)

(g) GO (h) PSO (i) eGO

Figure 3: The MS and Pan image of QuickBird and the results of image fusion using various methods

6 Conclusions

This paper employs the compact strategy to improve the original GO algorithm and introduces Lévy flight to improve the eGO algorithm. The individuals generated in enhanced GO are based on a probabilistic model called perturbation vector (PV), which occupies less memory than an actual population. Meanwhile, the enhanced GO algorithm retains the search mechanism of the original GO algorithm, making it suitable for devices with limited memory. Experimental results on the CEC2017 test function demonstrate that eGO is efficient and effective compared to GO and other compact algorithms. When applied to image fusion, the α_i found by eGO produces better fusion results compared to the previous PSO approach and other algorithms mentioned above. This has a beneficial effect on following image processing procedures.

Acknowledgement: The authors would like to express their gratitude to all the anonymous reviewers and the editorial team for their valuable feedback and suggestions.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Study conception and design: Jeng-Shyang Pan, Shu-Chuan Chu, Junzo Watada; data collection: Wenda Li, Xiao Sui, Jeng-Shyang Pan; analysis and interpretation of results: Wenda Li, Xiao Sui, Shu-Chuan Chu; draft manuscript preparation: Wenda Li, Xiao Sui, Junzo Watada. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The source code of the GO can be found at https://github.com/tsingke/Growth-Optimizer (accessed on 03 August 2022). The source code for the eGO can be obtained from the corresponding author upon request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Pierezan and L. Dos Santos Coelho, "Coyote optimization algorithm: A new metaheuristic for global optimization problems," in *2018 IEEE Congr. Evol. Computat. (CEC)*, Rio de Janeiro, Brazil, 2018, pp. 1–8. doi: 10.1109/CEC.2018.8477769.

[2] Q. Zhang, H. Gao, Z. -H. Zhan, J. Li, and H. Zhang, "Growth optimizer: A powerful metaheuristic algorithm for solving continuous and discrete global optimization problems," *Knowl.-Based Syst.*, vol. 261, 2023, Art. no. 110206. doi: 10.1016/j.knosys.2022.110206.

[3] H. B. Aribia *et al.*, "Growth optimizer for parameter identification of solar photovoltaic cells and modules," *Sustainability*, vol. 15, no. 10, 2023, Art. no. 7896. doi: 10.3390/su15107896.

[4] H. Gao, Q. Zhang, X. Bu, and H. Zhang, "Quadruple parameter adaptation growth optimizer with integrated distribution, confrontation, and balance features for optimization," *Expert. Syst. Appl.*, vol. 235, 2024, Art. no. 121218. doi: 10.1016/j.eswa.2023.121218.

[5] A. Fatani *et al.*, "Enhancing intrusion detection systems for IoT and cloud environments using a growth optimizer algorithm and conventional neural networks," *Sensors*, vol. 23, no. 9, 2023, Art. no. 4430. doi: 10.3390/s23094430.

[6] A. Kaveh and K. B. Hamedani, "A hybridization of growth optimizer and improved arithmetic optimization algorithm and its application to discrete structural optimization," *Comput. Struct.*, vol. 303, 2024, Art. no. 107496. doi: 10.1016/j.compstruc.2024.107496.

[7] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer Science & Business Media, 2001, vol. 2. Accessed: Jun. 10, 2024. [Online]. Available: https://dl.acm.org/doi/book/10.5555/559621

[8] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, 1999. doi: 10.1109/4235.797971.

[9] R. Rastegar and A. Hariri, "A step forward in studying the compact genetic algorithm," *Evol. Comput.*, vol. 14, no. 3, pp. 277–289, 2006. doi: 10.1162/evco.2006.14.3.277.

[10] G. R. Harik, F. G. Lobo, and K. Sastry, "Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA)," in *Scalable Optimization via Probabilistic Modeling*. Springer, 2006, pp. 39–61. Accessed: Jun. 10, 2024. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-540-34954-9_3.pdf

[11] K. Sastry, D. E. Goldberg, and D. Johnson, "Scalability of a hybrid extended compact genetic algorithm for ground state optimization of clusters," *Mater. Manuf. Process.*, vol. 22, no. 5, pp. 570–576, 2007. doi: 10.1080/10426910701319654.

[12] J. C. Gallagher and S. Vigraham, "A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks," in *Proc. 4th Ann. Conf. Genet. Evol. Computat.*, 2002, pp. 163–170. Accessed: Jun. 10, 2024. [Online]. Available: https://gpbib.cs.ucl.ac.uk/gecco2002/EH200.pdf

[13] F. Neri, E. Mininno, and G. Iacca, "Compact particle swarm optimization," *Inf. Sci.*, vol. 239, pp. 96–121, 2013. doi: 10.1016/j.ins.2013.03.026.

[14] C. T. Brown, L. S. Liebovitch, and R. Glendon, "Lévy flights in Dobe Ju/'hoansi foraging patterns," *Hum. Ecol.*, vol. 35, pp. 129–138, 2007. doi: 10.1007/s10745-006-9083-4.

[15] G. Wu, R. Mallipeddi, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC, 2017 competition and special session on constrained single objective real-parameter optimization," 2017. [Online]. Available: https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC2017/CEC2017.htm

[16] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: A comprehensive survey," *Artif. Intell. Rev.*, vol. 52, no. 4, pp. 2191–2233, 2019. doi: 10.1007/s10462-017-9605-z.

[17] R. B. Ash and C. A. Doléans-Dade, *Probability and Measure Theory*. Academic Press, 2000. Accessed: Jun. 10, 2024. [Online]. Available: https://www.colorado.edu/amath/sites/default/files/attached-files/billingsley.pdf

[18] W. J. Cody, "Rational Chebyshev approximations for the error function," *Math. Comput.*, vol. 23, no. 107, pp. 631–637, 1969. doi: 10.1090/S0025-5718-1969-0247736-4.

[19] P. Barthelemy, J. Bertolotti, and D. S. Wiersma, "A Lévy flight for light," *Nature*, vol. 453, no. 7194, pp. 495–498, 2008. doi: 10.1038/nature06948.

[20] A. F. Kamaruzaman, A. M. Zain, S. M. Yusuf, and A. Udin, "Levy flight algorithm for optimization problems-a literature review," *Appl. Mech. Mater.*, vol. 421, pp. 496–501, 2013. doi: 10.4028/www.scientific.net/AMM.421.496.

[21] J. -S. Pan, P. -C. Song, S. -C. Chu, and Y. -J. Peng, "Improved compact cuckoo search algorithm applied to location of drone logistics hub," *Mathematics*, vol. 8, no. 3, 2020, Art. no. 333. doi: 10.3390/math8030333.

[22] A. -Q. Tian, S. -C. Chu, J. -S. Pan, H. Cui, and W. -M. Zheng, "A compact pigeon-inspired optimization for maximum short-term generation mode in cascade hydroelectric power station," *Sustainability*, vol. 12, no. 3, 2020, Art. no. 767. doi: 10.3390/su12030767.

[23] L. Liu, M. Chu, R. Gong, L. Liu, and Y. Yang, "Weighted linear loss large margin distribution machine for pattern classification," *Chin. J. Electron.*, vol. 33, no. 3, pp. 753–765, 2024. doi: 10.23919/cje.2022.00.156.

[24] H. Wang *et al.*, "Detecting double mixed compressed images based on quaternion convolutional neural network," *Chin. J. Electron.*, vol. 33, no. 3, pp. 657–671, 2024. doi: 10.23919/cje.2022.00.179.

[25] M. Choi, "A new intensity-hue-saturation fusion approach to image fusion with a tradeoff parameter," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 6, pp. 1672–1682, 2006. doi: 10.1109/TGRS.2006.869923.

[26] H. R. Shahdoosti and H. Ghassemian, "Combining the spectral PCA and spatial PCA fusion methods by an optimal filter," *Inf. Fusion*, vol. 27, pp. 150–160, 2016. doi: 10.1016/j.inffus.2015.06.006.

[27] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Process. Lett.*, vol. 9, no. 3, pp. 81–84, 2002. doi: 10.1109/97.995823.

Appendix A. Comparison of Standard Deviation and Minimum Value of eGO, eGO_l, GO, and Commonly Used Compact Algorithms

Table A1: Standard deviation and minimum value on 30 dimensions of CEC2014

Func_Num	Std_cCS.	Min_cCS.	Std_cBA.	Min_cBA.	Std_cPIO.	Min_cPIO.	Std_cPSO.	Min_cPSO.	Std_cSCA.	Min_cSCA.	Std_GO.	Min_GO.	Std_eGO_l.	Min_eGO_l.	Std_eGO.	Min_eGO.
1	3.95E+08	1.53E+09	28225282	4.25E+08	3.86E+08	9.8E+08	5.22E+08	1.79E+09	2.31E+08	6.03E+08	6121457	8452682	10550131	17849786	10550131	4222181
2	1.53E+10	6.16E+10	1.39E+09	3.25E+10	5.57E+09	8.55E+10	1.01E+10	9.26E+10	5.47E+09	3.89E+10	68238998	43859550	99800102	1.14E+08	99800102	6022729
3	63577.28	174271	1226.105	79235.02	38973.63	147026.1	1.9E+08	1085237	34510.13	71246.88	3507.642	4745.418	2962.029	7241.961	2962.029	7466.123
4	5039.53	14934.31	207.49	4330.499	3938.414	8205.196	3454.002	19077.57	1383.962	3086.178	40.53111	473.1227	26.55192	483.7601	26.55192	429.9383
5	0.07441	521.0602	0.063482	520.3435	0.135571	520.6164	0.121875	521.4895	0.056783	520.9694	0.051669	520.9373	0.037779	521.0164	0.037779	520.8937
6	1.776198	638.1262	0.911426	647.7297	1.676046	641.3906	2.928122	646.4715	2.186988	635.1547	4.403064	618.0611	2.639239	618.9578	2.639239	617.8477
7	98.77281	1513.245	14.90414	1051.672	89.36666	1388.606	109.4232	1620.21	67.64058	976.5147	0.764841	701.2239	1.946154	705.2904	1.946154	701.1221
8	23.84832	1221.526	4.007215	953.3698	6.410413	1185.876	40.60377	1159.031	27.42542	1077.705	14.64169	880.5801	16.84708	990.3625	16.84708	864.9838
9	35.87384	1384.401	4.654058	1084.228	1.936777	1343.673	44.48819	1328.318	26.90002	1201.341	16.77814	1068.735	18.57018	1129.925	18.57018	1010.497
10	308.1519	9166.071	163.0366	5172.353	311.4695	8835.563	1040.593	9887.49	354.5176	7921.634	575.15	3695.057	824.2536	4550.882	824.2536	2413.546
11	333.4004	8347.856	337.9939	5366.811	314.1249	9141.391	1192.493	11214.24	327.2936	8547.742	359.3324	7585.523	480.1648	6548.492	480.1648	3540.768
12	0.609782	1202.952	1.069302	1201.132	0.397097	1203.178	4.628009	1208.441	0.572988	1202.4	0.456264	1202.108	0.43105	1202.276	0.43105	1201.283
13	0.953906	1307.802	0.097031	1305.627	0.769945	1306.725	0.67123	1309.701	0.595621	1304.378	0.097527	1300.439	0.130435	1300.607	0.130435	1300.516
14	46.17396	1630.116	6.225484	1546.538	20.42087	1633.934	27.61834	1786.861	18.45719	1512.247	0.125725	1400.241	0.394617	1400.549	0.394617	1400.262
15	3370345	4802485	436.1691	3043.242	7856.403	915773.7	1746781	1183078	145204	68457.37	4.622933	1520.37	298.7756	1563.514	298.7756	1516.182
16	0.196172	1613.3	0.479692	1613.742	0.176505	1613.373	0.362779	1614.108	0.266954	1612.713	0.242499	1612.754	0.328581	1611.758	0.328581	1612.041
17	53184230	80054704	4681854	34670432	36501021	51936936	3.73E+08	6.43E+08	14937133	8521356	349371	136459.5	1918069	3434364	1918069	234542.2
18	1.99E+09	1.7E+09	1.95E+08	4.27E+08	1.34E+09	1.58E+09	2.34E+09	1.2E+10	6.8E+08	2.87E+08	48958.79	4318.715	741295.7	21309.93	741295.7	6212.983
19	165.6867	2421.465	9.495968	2127.223	134.9974	2221.574	234.3156	2718.727	56.22401	2021.983	17.52816	1909.49	15.15849	1918.43	15.15849	1912.083
20	1596147	1080680	15477.76	127534.9	703448.5	378385	1.73E+09	1.21E+09	215324.3	29738.69	2745.385	2648.054	6525.656	16452.54	6525.656	14067.73
21	24240512	27155284	2242585	13702441	22987372	12546700	1.15E+09	1.15E+09	7006907	2691403	46635.38	33421.82	1370889	923546.5	1370889	41504.35
22	431.1423	3280.318	702.2986	4060.413	487.7472	3516.289	6623530	1389791	225.806	3336.36	125.6472	2646.643	170.2381	2729.722	170.2381	2601.612
23	330.2067	3091.394	0.825075	2510.654	0.476924	2502.334	200.9618	3397.75	58.44381	2732.113	1.140975	2615.909	6.27082	2569.641	6.27082	2570.466
24	25.8612	2832.065	1.21678	2602.186	0.137552	2600.734	32.5636	2670.424	19.34881	2607.333	5.8796	2629.928	0.741944	2608.667	0.741944	2606.165
25	24.94927	2803.892	0.013715	2700.17	0.007354	2700.024	22.25226	2721.18	14.94456	2705.338	2.16477	2707.997	0.092876	2701.186	0.092876	2701.12
26	59.04913	2705.379	0.001389	2800.004	20.22504	2709.229	18.56593	2811.507	32.03173	2703.781	0.103792	2700.402	22.17158	2700.989	22.17158	2700.66
27	101.3428	4067.928	2.038334	2909.249	0.391628	2900.875	2590.987	19410.97	262.0358	3344.205	166.4921	3115.886	149.3179	3107.779	149.3179	3104.498
28	813.7812	7795.584	3.799886	3013.036	0.44218	3001.226	2194.178	17257.67	412.1718	5890.612	114.7493	3770.279	103.9753	3334.006	103.9753	3355.392
29	91466061	65318012	1151024	12651603	484296.9	2895973	3.85E+08	1.1E+09	18580104	38262175	1936268	6596.831	28242735	79554.3	28242735	3107.697
30	1379253	1494938	89759.61	779710.8	36597.21	182099.3	26357613	73623122	359728	344525.1	2621.942	6638.634	1887133	58644.07	1887133	3438.418
win	24	30	15	23	15	24	29	30	24	30	10	21	30	26		

Table A2: Standard deviation and minimum value on 50 dimensions of CEC2014

Func_Num	Std_cCS	Min_cCS	Std_cA	Min_cBA	Std_cPIO	Min_cPIO	Std_cPSO	Min_cPSO	Std_cSCA	Min_cSCA	Std_GO	Min_GO	Std_eGO_1	Min_eGO_1	Std_eGO	Min_eGO
1	1.24E+09	3.68E+09	2.15E+08	2.18E+09	9.95E+08	2.74E+09	2.2E+09	1.28E+10	7.3E+08	1.06E+09	49913001	49653496	18894604	72729521	5886285	10359331
2	2.5E+10	2.08E+11	2.35E+09	8.77E+10	1.32E+09	1.92E+11	1.63E+10	1.77E+11	1.26E+10	8.27E+10	6.4E+08	8.98E+08	5.29E+08	1.06E+09	30175581	53528881
3	33660.3	265707.8	3169.943	137574.8	45212.93	306931.1	5.48E+08	18937054	45012.76	166244.7	14607	61530.78	8610.228	31119.81	5862.024	29483.45
4	15553.57	40690.02	835.1063	17300.8	5998.941	52875.18	8584.553	67371.37	5050.115	17490.2	152.2073	775.9688	89.54381	843.5166	31.28311	523.309
5	0.040546	521.2131	0.059669	520.625	0.091384	520.9246	0.094564	521.4632	0.042746	521.1889	0.033269	521.176	0.049407	521.0946	0.041072	521.1462
6	1.752692	677.7424	1.001111	683.2623	1.595598	675.7423	3.643913	682.1655	2.043766	671.0141	5.871215	642.8761	2.646772	643.1951	5.422745	635.2495
7	194.0592	2638.088	29.75155	1493.758	18.94765	2395.621	105.2351	2504.471	67.34477	1652.05	6.492346	712.8794	4.680323	717.7622	0.278829	701.8299
8	35.45632	1633.704	5.327054	1094.032	6.421397	1520.134	52.15398	1536.967	34.86731	1388.573	32.75298	1040.134	25.86431	1230.868	37.83437	994.3861
9	40.89856	1972.198	8.191744	1322.785	14.00558	1814.298	50.33909	1780.771	41.80722	1514.557	18.5937	1304.765	38.37552	1320.029	40.90468	1157.408
10	528.3859	15274.5	241.3524	8972.442	377.2008	15229.05	1422.824	17546.16	493.8687	13936.58	649.3228	9719.557	797.3124	11148.74	877.3896	4466.354
11	401.6286	15949.31	324.7916	9790.288	374.9956	15533.74	1132.66	19223.98	374.3816	14925.43	548.2928	13719.05	766.4565	12644.67	562.0279	8201.841
12	0.611502	1203.175	1.604238	1203.21	0.520703	1204.181	2.778904	1209.17	0.380993	1204.159	0.432003	1203.353	0.594164	1203.112	0.410426	1201.768
13	0.842572	1308.834	0.105449	1306.03	0.03576	1309.49	0.506462	1309.447	0.539699	1305.893	0.095676	1300.553	0.124811	1300.85	0.070179	1300.564
14	52.58531	1872.662	7.821169	1592.31	1.680976	1779.351	35.23125	1848.444	45.78352	1592.128	2.322924	1400.376	3.224051	1410.697	0.127903	1400.308
15	26239704	36604028	36901.42	190528.9	1408621	18039244	21502465	8539637	999274.4	1123343	428.709	1573.541	9727.417	6264.391	9.433713	1547.061
16	0.174282	1623.246	0.765847	1622.537	0.127838	1622.874	0.48916	1623.918	0.274525	1622.625	0.307472	1622.279	0.216613	1622.168	0.444565	1621.348
17	2.28E+08	4.03E+08	29394585	3.01E+08	1.7E+08	2.97E+08	9.03E+08	2.61E+09	91534906	87044475	2058692	1094479	6893775	11670444	856648.1	511104.4
18	4.78E+09	1.45E+10	4.73E+08	8.04E+09	3.67E+09	8.63E+09	3.25E+09	3.2E+10	1.07E+09	3.43E+09	127387.5	40912.95	11093656	3148356	429862.9	414796.2
19	764.1479	3967.24	53.97698	2627.438	609.7892	3336.038	1974.205	7871.177	177.3393	2328.304	21.86324	1927.282	7.455915	1937.904	3.301421	1921.198
20	3189700	378924.8	10908.94	89935.29	1423061	386304.1	2.15E+09	6.03E+08	366750.4	115034.6	10085.39	9404.825	7804.215	20982.96	6019.197	18184.79
21	99406845	1.18E+08	2447134	6991671	67009113	47796714	6.84E+08	5.73E+08	21065938	21356785	815705.4	301528.9	1442760	3010875	471265	768575.5
22	86018.21	17623.83	6174.729	19357.19	41082.85	9547.843	3044938	3280020	1042.103	4761.44	237.5945	3569.755	539.2647	3452.228	302.3084	3078.98
23	408.9575	4282.653	1.87932	2522.538	0.5101	2505.593	220.9272	3822.933	313.716	3435.407	5.114597	2649.925	6.267613	2603.97	8.10891	2605.957
24	41.95601	3139.034	2.637768	2606.688	0.238322	2601.951	40.89972	2785.272	41.39245	2732.416	4.415694	2692.272	1.94565	2612.412	1.587069	2613.092
25	55.03897	3010.074	0.041487	2700.503	0.014605	2700.122	29.48787	2741.843	12.84813	2737.51	6.436844	2727.44	0.138123	2702.479	0.138779	2702.656
26	154.7144	2887.02	0.00343	2800.017	17.1584	2745.662	32.99318	2820.853	22.11289	2804.883	54.4614	2700.566	0.116472	2800.099	22.26077	2700.616
27	88.2868	5289.507	3.581322	2936.004	0.755208	2901.311	3250.162	34771.88	92.67701	4887.246	158.0482	3960.866	176.0978	3158.832	198.9132	3123.494
28	541.4272	14871.13	8.195719	3065.69	1.324827	3008.986	3827.294	28015.12	1093.443	9928.67	468.6972	4465.687	124.3857	3639.39	104.5148	3721.888
29	2.63E+08	1.25E+09	2954787	40731471	1251018	9977575	4.19E+08	3E+09	1.11E+08	3.4E+08	15529022	52966.45	3591368	933478.6	41039409	3110.946
30	10584956	24932523	124402.8	1841271	42917.41	438490.4	24781499	1.01E+08	2986250	3411561	13059.59	32932.63	1811963	194678.7	1984192	3419.068
win	22	30	17	24	15	24	29	30	22	30	18	25	19	25		

Table A3: Standard deviation and minimum value on 100 dimensions of CEC2014

Func_Num	Std_cCS	Min_cCS	Std_cBA	Min_cBA	Std_cPIO	Min_cPIO	Std_cPSO	Min_cPSO	Std_cSCA	Min_cSCA	Std_GO	Min_GO	Std_eGO_l	Min_eGO_l	Std_eGO	Min_eGO
1	2.36E+09	1.18E+10	1.82E+08	3E+09	1.34E+09	1.29E+10	2.42E+09	1.48E+10	1.42E+09	3.38E+09	1.12E+08	4.05E+08	55750856	2.39E+08	19833984	95116359
2	3.83E+10	4.95E+11	4.51E+09	1.88E+11	2.54E+08	3.37E+11	1.81E+10	3.13E+11	1.85E+10	2.26E+11	6.36E+09	2.57E+10	2E+09	6.71E+09	2.52E+08	3.32E+08
3	74120	605508.4	3760.71	290437.4	81670.08	531974.3	2.03E+08	18325393	57421.48	408089.3	38888.93	230227.2	14920.8	143679.8	19284.14	98265.75
4	30357.46	159928.5	2271.823	41115.26	356.6399	137871.8	15771.29	126619.5	14784.8	44405.5	653.1846	2788.528	332.9177	1712.388	60.23336	755.2291
5	0.032702	521.3677	0.049433	520.9761	0.062603	521.1737	0.057434	521.6063	0.033675	521.3412	0.027397	521.3632	0.029398	521.3374	0.01852	521.3553
6	2.12626	763.7775	1.565458	769.7065	0.555513	768.4154	5.37163	768.371	3.947471	755.832	10.54403	705.0484	4.405346	715.0697	4.944779	708.0298
7	327.7586	5355.369	25.93617	2714.415	2.808177	4130.178	168.6425	3887.741	226.1575	2977.905	44.21676	945.9812	25.55194	767.0994	1.615291	705.8811
8	70.53041	2595.655	9.760394	1391.345	20.95852	2275.712	73.23272	2298.023	52.88445	2154.2	37.66706	1603.947	32.95907	1953.646	64.76506	1380.354
9	98.4569	3083.855	15.69857	1663.458	3.384051	2544.103	76.77033	2613.522	64.37217	2338.43	40.22527	1941.093	64.98861	2009.556	69.28756	1757.816
10	775.1353	33022.43	329.8764	17894.11	808.0349	32748.68	1797.254	36067.42	1021.03	30688.11	1104.94	23923.71	712.579	28525.58	1438.628	15529.51
11	403.4937	34343.91	611.8615	14699.49	604.1518	32797.14	1527.893	36376.05	468.6005	33042.94	693.9472	31257.85	635.2223	29584.74	1093.275	19907.21
12	0.490556	1204.645	0.693719	1203.206	0.408685	1204.98	1.418516	1209.02	0.33332	1204.475	0.314571	1204.29	0.358945	1204.418	0.343562	1202.591
13	0.521609	1312.343	0.081868	1307.448	0.009403	1310.122	0.372305	1309.835	0.379209	1308.295	0.728245	1300.94	0.625369	1301.173	0.088639	1300.635
14	84.35805	2823.481	14.13074	1986.399	1.957071	2426.927	38.69201	2413.264	50.09802	2105.402	15.9096	1470.575	10.40032	1424.384	0.325817	1400.445
15	92249067	2.1E+08	138763.1	2374004	319607.5	55682519	39428452	33005065	5428067	7034011	15448.72	9791.902	63591.34	125793	38.8697	1637.1
16	0.211089	1647.714	0.364804	1646.372	0.145985	1647.416	0.6313	1648.569	0.29777	1647.062	0.204127	1647.332	0.245064	1646.569	0.32765	1646.168
17	3.97E+08	1.31E+09	45947298	5.77E+08	5.04E+08	1.41E+09	7.12E+08	2.83E+09	2.24E+08	3.55E+08	14330758	33586456	14894520	24399428	3839684	6640986
18	9.38E+09	4.51E+10	9.24E+08	1.86E+10	3.62E+09	4.13E+10	5.28E+09	5.35E+10	5.33E+09	1.16E+10	16982330	10869854	53986090	28150809	2827929	4792195
19	3258.828	10572.35	230.5468	4838.078	2313.909	9378.747	3337.424	14459.06	947.1187	4239.156	37.4703	2112.76	24.54962	2052.878	32.67822	1969.007
20	6762650	9917121	40782.73	468464	5768408	6525885	8.71E+08	58021096	1678843	362268.8	115660.2	66016.32	34664.8	101870.4	19861.4	80458.43
21	2.77E+08	7.46E+08	12358353	94174137	1.85E+08	7.46E+08	5.14E+08	9.61E+08	1.47E+08	2.07E+08	5368634	11637860	3662788	7922889	1292280	3743573
22	455084	106402.2	2484.038	12791.17	270717.1	174114.9	1569092	639885.1	4096.761	8643.589	249.8665	6641.858	538.1361	4380.821	517.755	4286.574
23	719.3952	7784.163	1.737339	2534.586	0.664336	2506.738	193.9216	3841.311	149.3108	3769.398	15.99589	2640.649	4.705399	2601.305	4.490934	2600.432
24	16.11909	2620.3	0.546491	2608.826	0.159326	2601.813	89.92269	3027.975	18.15642	2555.936	0.082193	2500.412	0.174498	2500.586	0.890174	2500
25	1.995133	2678.631	0.042705	2698.376	1.75349	2678.555	0.42431	2694.745	2.80054	2662.961	1.127946	2612.466	0.367326	2612.204	0.794165	2602.418
26	49.51496	2607.125	3.962684	2763.95	23.5179	2712.345	0.001128	2800.001	40.07957	2600.035	44.42617	2600	30.77935	2600	8.6E-13	2600
27	0.427803	3000.152	0.004602	2900.077	0.0013	2900.015	0.78018	2904.461	0.665694	2904.21	3.75E-06	3000	0.010078	2900.231	0.009965	2900.231
28	2.582056	3090.506	0.040289	3000.631	0.012166	3000.119	2.340337	3026.679	1.806135	3026.499	1.583208	3023.461	0.108846	3001.849	0.648234	3000
29	1.430361	3078.315	0.053031	3097.668	1.799238	3074.008	0.651397	3091.895	2.960061	3057.121	1.334233	3014.704	0.526885	3015.895	1.047072	3003.322
30	2.048126	3179.391	0.419682	3190.995	2.788317	3175.272	0.395545	3194.272	2.302262	3163.661	1.121324	3112.611	0.422439	3112.517	1.289854	3100
win	26	30	16	25	19	27	27	30	23	29	20	28	17	29		

Table A4: Standard deviation and minimum value on 30 dimensions of CEC2017

Func_Num	Std_cCS	Min_cCS	Std_cBA	Min_cBA	Std_cPIO	Min_cPIO	Std_cPSO	Min_cPSO	Std_cSCA	Min_cSCA	Std_GO	Min_GO	Std_eGO_l	Min_eGO_l	Std_eGO	Min_eGO
1	1.22E+10	6.77E+10	1.09E+09	2.32E+10	6.27E+09	5.82E+10	7.79E+09	7.46E+10	6.09E+09	2.2E+10	47506214	37781504	92190685	81470731	3428101	7681031
2	1.99E+46	2.29E+39	2.75E+48	8.69E+47	9.38E+46	2.2E+40	4.36E+62	2.04E+59	5.86E+40	1.17E+36	3.9E+29	9.01E+21	4.17E+32	1.5E+24	5.22E+15	5.21E+11
3	401627.7	170035.4	1737.891	63963.96	58043.18	132390.2	4.76E+12	101113.5	37716.89	87259.62	25570.77	27704.02	6658.387	59773.15	8260.43	19125.73
4	6880.939	15081.86	327.6222	5583.425	4053.231	10310.71	7093.705	32080.39	2399.843	1932.686	29.37732	518.7691	18.53182	481.849	15.18471	423.6571
5	61.59068	923.8786	5.806804	788.7986	28.5446	933.7517	52.64164	1019.387	31.74284	817.5309	18.20755	654.7898	19.64446	714.3635	27.0885	624.7552
6	9.296535	702.3377	1.965656	664.5042	6.010065	690.0371	15.60257	718.2452	9.929785	667.6709	2.524453	604.5702	10.7333	622.3926	7.856777	628.7091
7	135.424	2594.306	16.97192	1318.532	8.510227	1548.269	102.8976	1590.29	59.58643	1290.739	23.16359	922.0628	20.44676	986.3328	22.62385	952.0854
8	34.07735	1273.24	4.964285	986.3269	8.470024	1227.334	46.44071	1229.352	20.80369	1107.268	16.38808	966.0264	19.09036	994.2034	22.43653	903.6179
9	3988.672	18593.32	258.1879	5599.271	801.4718	17867.93	4884.902	18392.67	2408.853	8200.262	468.7858	1196.22	936.5468	4688.884	1099.849	3176.607
10	361.8944	9354.061	306.2013	5019.795	336.7442	9145.04	1130.126	10394.74	269.0601	8761.595	353.643	7868.788	417.6428	6667.716	350.6517	4364.691
11	7404.713	9005.997	235.816	2929.29	4875.97	11471.05	1.22E+09	1.43E+08	1590.043	3199.799	38.38703	1274.417	718.7439	2119.564	29.50886	1156.739
12	3.3E+09	9.85E+09	4.75E+08	5.33E+09	2.9E+09	8.89E+09	4.1E+09	2.3E+10	1.82E+09	2.12E+09	1805222	1352536	20903361	7578444	2676714	1894439
13	3.8E+09	4.57E+09	6.4E+08	3.66E+09	3.7E+09	9.73E+08	6.47E+09	3.02E+10	1.16E+09	7.13E+08	206166.6	12193.2	5922001	3046398	150169.1	118531.2
14	5833111	798489.4	126008.9	952661.4	5482747	1062769	8.44E+08	7.87E+08	1523075	290742.5	8256.805	2541.081	253962.5	508312.1	74447.76	8300.969
15	1.31E+09	5.8E+08	1561.333	14739.2	7.99E+08	4.19E+08	2.19E+09	5.3E+09	78751916	26254357	46245.45	15093.86	43285.91	7596.033	8828.801	4790.258
16	382.278	5147.796	208.5034	5460.113	613.5021	4759.472	6005.421	21250.54	275.7692	4255.291	211.4476	2784.979	238.1409	2630.687	265.5342	2632.775
17	957.1164	3733.817	99.50581	2914.172	389.6266	3038.26	389016.5	82594.77	205.8992	2815.635	163.0755	1969.639	233.1751	1922.307	340.7242	1813.738
18	65225893	18174401	1119438	1908465	68977365	62310569	1.91E+09	1.8E+09	19511229	2042105	226947	78471.53	842445.8	863682	191302.5	44093.3
19	8.76E+08	1.78E+09	267044	771963.4	9.25E+08	9.65E+08	2.61E+09	3.92E+09	1.17E+08	30861443	66370.2	6730.301	55412.94	10709.35	16417.2	6740.715
20	152.681	3140.432	197.4005	3109.028	117.1748	3091.95	598.2847	3520.163	113.8094	2887.167	102.7175	2523.642	167.4024	2425.633	132.8087	2386.519
21	41.83007	2730.831	20.7569	2690.085	32.84323	2714.733	153.592	3021.451	34.14593	2615.727	8.612884	2499.763	18.83851	2514.007	68.08341	2207.261
22	792.6322	8705.108	209.9887	7841.984	588.4043	8925.126	965.0486	11013.78	1372.509	6216.522	3281.861	2336.933	4.944331	2334.711	1.658195	2321.294
23	67.83131	3191.238	24.57121	4218.339	91.07931	3332.539	466.9201	6722.285	42.56611	3069.476	20.40967	2831.674	37.29122	2879.488	59.75274	2843.819
24	194.3297	3394.855	32.97441	4180.916	112.2384	3581.964	111.037	5073.635	38.1855	3268.66	18.04042	3000.89	29.44765	3009.274	36.81253	2947.712
25	2117.007	8595.249	44.38789	3329.501	537.9006	7105.868	1512.512	8373.069	614.6	3633.439	18.38265	2894.467	19.57378	2910.53	14.04401	2881.163
26	722.9798	9590.317	276.1142	8997.228	760.7056	11565.53	1803.721	14972.18	399.6404	7973.926	557.7409	3624.309	1379.809	3129.503	1605.968	2895.911
27	277.3935	3948.682	115.0267	6691.956	227.2155	3800.919	799.5561	9300.112	95.92089	3553.872	12.97251	3198.654	20.64198	3249.215	18.96111	3179.66
28	969.6361	9070.288	81.87084	4885.17	641.4522	7962.767	1172.248	9453.044	516.5781	4321.322	32.19989	3241.121	24.42944	3266.576	22.10366	3206.885
29	657.4534	6038.079	221.6944	6536.824	805.8345	5506.702	803337.2	97695.73	560.8549	4726.391	197.5769	3871.75	268.3367	3908.545	179.2677	3780.701
30	5.88E+08	8.07E+08	92037048	2.01E+08	4.87E+08	4.55E+08	1.61E+09	7.23E+09	1.81E+08	1.05E+08	166683.2	70668.98	963587.9	589024	249537.6	104775.5
win	28	30	16	30	22	30	30	30	23	30	15	21	19	28		

Table A5: Standard deviation and minimum value on 50 dimensions of CEC2017

Func_Num	Std_cCS	Min_cCS	Std_cBA	Min_cBA	Std_cPIO	Min_cPIO	Std_cPSO	Min_cPSO	Std_cSCA	Min_cSCA	Std_GO	Min_GO	Std_eGO_l	Min_eGO_l	Std_eGO	Min_eGO
1	1.75E+10	1.75E+11	2.04E+09	5.4E+10	1.32E+08	1.34E+11	8.05E+09	1.29E+11	1.16E+10	7.15E+10	7.06E+08	1.15E+09	3.22E+08	9.33E+08	23781590	46516691
2	1.47E+86	8.98E+77	1.26E+69	1.02E+66	2.24E+84	1.05E+78	2.28E+96	1.21E+86	7.29E+76	1.75E+67	6.56E+57	2.91E+45	4.01E+60	5.17E+49	1.44E+36	5.65E+25
3	2516750	346632.7	4077.887	155261.7	212411.3	319045.1	3.04E+14	6.99E+09	72130.96	195868.8	42647.01	112927.2	17368.24	122980.8	16533.16	93361.7
4	9492.432	57427.85	656.4516	14096.96	3142.232	44816.68	7507.522	50892.52	4382.961	11881.29	102.3616	697.2635	66.01919	674.3561	34.82507	514.2527
5	55.57306	1432.431	10.75453	868.1947	11.61649	1297.753	68.9668	1290.194	45.98001	1117.815	36.55025	869.5868	30.33397	950.2542	26.13229	783.3619
6	6.018576	724.3366	1.410012	668.1554	3.08177	704.3837	10.31378	727.1964	7.064108	684.8846	3.54657	618.1959	9.75088	635.6717	5.515419	655.1453
7	278.6951	4779.255	17.17962	1767.29	7.824255	2093.943	122.9984	2338.315	129.1624	1848.813	32.79435	1189.07	34.49184	1294.932	57.46762	1246.823
8	53.94422	1789.543	6.597306	1213.774	1.800718	1675.525	66.70373	1637.98	30.77964	1449.902	31.41252	1184.734	29.16269	1249.522	41.17579	1080.739
9	8397.754	61069.85	664.9135	15258.61	2128.069	42851.81	8408.333	63115.86	6219.45	28970.87	3263.261	3507.292	3632.871	17783.99	5484.428	14448.8
10	777.2863	14327.41	1019.204	8642.72	395.147	15738.28	1532.972	16258.39	495.7417	14556.44	431.5582	14196.42	513.6263	12848.79	646.9493	7589.583
11	13072.48	39141.77	555.2928	9018.448	6783.725	37309.41	2.57E+09	28084.28	5600.209	10125.08	602.7306	2174.862	1392.167	4953.285	68.41712	1339.979
12	1.56E+10	6.91E+10	1.56E+09	4.42E+10	1.24E+10	5.74E+10	1.09E+10	1.29E+11	9.23E+09	2.23E+10	61130909	54473953	1.46E+08	2.14E+08	13624017	23362349
13	9.2E+09	2.27E+10	1.62E+09	2.27E+10	1.1E+10	1.64E+10	9.77E+09	1.01E+11	4.35E+09	3.15E+09	576386	124185.3	33034630	11285316	635560.9	1198263
14	26180900	18986627	7867623	48019240	24198816	42151753	3.83E+08	8.83E+08	8012695	4370095	110461.1	49418.42	2065843	2196085	160060	97027.5
15	2.98E+09	1.16E+10	1.77E+08	4.32E+08	3.45E+09	6.81E+09	4.03E+09	2.28E+10	7.72E+08	5.72E+08	225643.5	22874.25	3265367	187118.3	156896.7	85781.47
16	1232.774	7824.129	437.9856	5904.656	949.7827	7681.173	3346.755	18077.63	376.5904	6122.242	490.688	4038.718	418.7661	2739.438	438.0671	2783.136
17	128279.3	28173.76	288.5307	3126.961	38071.56	14240.01	248705.3	47796.47	445.0223	5086.651	203.8997	3598.11	322.7412	2565.98	340.1041	2654.032
18	1.46E+08	2.18E+08	3576770	27814209	1.1E+08	1.21E+08	8.08E+08	9.46E+08	33827458	24162846	1262813	392856.5	4366874	2492095	523789.5	1235346
19	1.86E+09	4.07E+09	1.07E+08	2.13E+08	1.45E+09	2.55E+09	3.16E+09	1.1E+10	4.01E+08	4.36E+08	60150.17	28517.74	996665	92802.69	47924.33	79597.96
20	239.8443	4315.227	122.9147	3702.57	169.4789	4570.007	504.1509	5599.458	172.1741	4081.669	220.9996	3643.151	346.3964	2833.056	188.6396	2778.226
21	54.39815	3225.853	26.93464	2932.083	43.35072	3227.836	265.9954	3980.696	48.36419	2993.618	31.55693	2656.877	25.55726	2748.219	55.5227	2545.783
22	711.0975	15951.02	260.4269	13354.44	495.5962	17280.02	1026.624	20350.75	476.6934	16389.15	2032.016	7738.779	5918.649	2352.34	3726.986	2345.504
23	119.4387	3893.555	26.6263	4886.225	85.64307	4381.664	582.6931	8014.291	102.3192	3600.633	39.37114	3140.69	69.94133	3257.293	126.2555	3171.43
24	279.1216	4031.229	41.71673	5604.038	195.9812	4398.757	171.3448	6603.649	112.6278	3834.061	33.4573	3313.829	43.23116	3214.44	68.79059	3165.087
25	4040.278	36853.99	239.0593	7726.001	58.95431	19473.27	1980.998	17938.34	1809.57	9026.067	81.66625	3241.202	32.05813	3136.066	46.2811	2944.376
26	1110.803	18551.42	275.7093	12891.7	55.85493	19752.02	1550.509	19025.81	1321.946	11945.48	456.5026	7650.424	2263.324	4178.623	1584.704	2952.864
27	383.8534	6725.209	471.2647	11212.32	509.1469	5772.631	1499.034	15495.56	300.3271	4664.405	71.62777	3361.451	128.7335	3542.919	178.4226	3200.011
28	2493.092	12773.38	196.0344	8468.086	1298.354	16251.06	3904.919	16627.4	1101.619	8047.862	248.8255	3479.018	92.95735	3494.177	27.42981	3282.916
29	106453.2	13900.96	2499.751	20429.72	52288.27	16015.23	13061036	1351683	3000.236	9001.047	350.083	4565.238	475.3141	4567.179	314.8909	4308.668
30	2.63E+09	4.72E+09	2.7E+08	1.65E+09	2.19E+09	3.89E+09	3.42E+09	2.01E+10	6.02E+08	1.54E+09	7265818	11782251	15993640	26430036	6576150	1673994
win	26	30	16	30	19	30	28	30	22	30	16	21	19	27		

Table A6: Standard deviation and minimum value on 100 dimensions of CEC2017

Func_Num	Std_cCS	Min_cCS	Std_cBA	Min_cBA	Std_cPIO	Min_cPIO	Std_cPSO	Min_cPSO	Std_cSCA	Min_cSCA	Std_GO	Min_GO	Std_eGO_l	Min_eGO_l	Std_eGO	Min_eGO
1	2.49E+10	5.05E+11	3.64E+09	1.66E+11	3.41E+08	2.94E+11	1.77E+10	2.8E+11	1.95E+10	2.23E+11	4.64E+09	2.64E+10	1.99E+09	5.9E+09	1.87E+08	4.1E+08
2	65535	8.9E+172	2.5E+153	1.8E+151	65535	8.2E+168	65535	1.4E+184	65535	1.5E+159	4.8E+141	7.7E+127	1.4E+147	8.9E+136	6.7E+111	3.18E+88
3	1.71E+08	907874.3	3094.496	338410.2	2502632	789636.8	1.02E+16	13729398	298266.9	424191.7	60996.66	417036.8	32795.23	296204.6	29680.2	269448.3
4	27733.47	171128	1881.188	48578.54	194.5274	156193.7	16603.06	144353.6	8684.081	49590.04	960.3489	2900.574	400.6547	1859.335	79.13727	812.8562
5	90.36258	2728.272	14.40065	1350.132	4.422812	2313.337	87.0872	2305.786	90.51154	2017.976	47.0716	1563.275	71.17787	1616.549	56.27171	1417.903
6	5.103121	744.3637	1.595276	668.4242	1.901643	717.8343	8.58524	724.0671	7.027158	703.5027	5.645903	643.8869	6.560096	667.6004	3.293294	672.4218
7	549.9731	11082.85	33.62255	3309.202	12.89408	4157.377	235.3282	4312.696	233.718	4024.861	123.8382	2148.402	96.34226	2183.488	88.07019	2358.1
8	89.44364	3153.343	11.78044	1887.853	5.016359	2764.856	90.01952	2764.418	85.70924	2421.775	35.08523	1914.202	68.36152	1985.497	113.2334	1712.51
9	14785.57	150575.4	1956.597	30963.08	2252.186	74427.6	16125.76	87906.99	12697.69	86619.01	10063.03	31058.65	4846.97	58947.04	7583.954	57351.67
10	552.1519	33622.19	484.0421	17055.72	598.7616	33088.16	2193.066	36393.02	467.3727	32895.5	735.6213	31242.94	744.8319	29686.16	1127.133	18773.88
11	167601	428321.3	5405.617	144186.2	94164.61	333389.7	2.23E+13	5.84E+12	37761.49	221020.6	28460.05	73256.14	16484.91	64409.16	7917.016	16192.61
12	1.8E+10	2.61E+11	5.03E+09	1.13E+11	1.35E+10	2.16E+11	1.52E+10	2.49E+11	1.4E+10	9.48E+10	1.13E+09	1.54E+09	7.74E+08	1.6E+09	1.17E+08	1.6E+08
13	1.25E+10	4.09E+10	9.14E+08	2.27E+10	4.45E+09	5.16E+10	7.06E+09	5.9E+10	3.89E+09	1.8E+10	24675183	11065431	77163242	53473784	2538710	5140869
14	73366022	1.83E+08	693135	8431342	1.04E+08	74023873	6.8E+08	4.96E+08	44471713	40408204	1651619	2035821	1486201	5093339	683426.5	1521658
15	6.71E+09	2.38E+10	9.16E+08	8.99E+09	4.8E+09	1.76E+10	3.65E+09	3.67E+10	2.45E+09	5.52E+09	1124196	437628.4	14531407	5282790	552027.5	1175632
16	2988.982	24961.58	426.9567	14320.53	2030.892	24931.27	3499.637	34465.26	1007.023	14837.19	507.8886	9650.471	895.1711	6265.397	565.7006	6174.179
17	11335584	5086567	172426	508629.2	6527364	2543821	1.09E+08	1.14E+08	289492.2	22253.3	401.566	7047.644	9563.796	7436.542	483.295	4629.907
18	1.42E+08	2.34E+08	1241917	5581794	1.65E+08	2.82E+08	5.32E+08	1E+09	55785277	73641378	4409056	4479172	1969391	3677655	800600.1	1500290
19	5.22E+09	2.87E+10	6.89E+08	1E+10	4.88E+09	2.35E+10	4.13E+09	3.63E+10	1.7E+09	4.69E+09	1553666	749972.3	11708694	10930359	816352.8	993764.1
20	277.4527	7841.858	558.6038	6236.769	241.8314	8407.704	1186.255	9621.825	282.6375	7902.731	390.7954	7114.726	354.0549	6495.952	407.5146	4916.306
21	4.659656	2270.031	0.986059	2246.329	3.970196	2265.338	2.896501	2277.416	5.638501	2225.285	0.087054	2200.749	0.012098	2200.6	0.004934	2200.405
22	4.797874	2368.92	1.348903	2350.988	5.080781	2364.167	2.083401	2380.699	5.78376	2324.336	0.101682	2300.646	0.012893	2300.495	0.00476	2300.321
23	3.457022	2543.948	1.486659	2509.677	2.194135	2536.835	3.05588	2542.45	5.081665	2515.194	42.92062	2408.752	0.312482	2408.684	0.464828	2401.614
24	2.42201	2645.461	1.271291	2614.622	2.224066	2637.688	2.996158	2642.307	5.610044	2610.732	34.8491	2506.993	0.289642	2506.532	0.511886	2501.353
25	3.879834	2813.644	1.045117	2788.156	1.279345	2809.749	2.685309	2812.292	5.039585	2778.773	0.92741	2716.398	17.77366	2638.402	21.98548	2606.958
26	3.205589	2936.022	2.392036	2853.181	0.466249	2900.534	3.749794	2907.216	13.34591	2846.775	37.27658	2717.017	0.579209	2714.613	0.53327	2702.774
27	2.975941	3090.887	0.951746	3050.155	0.033334	3079.545	1.861311	3079.647	5.433862	3046.314	4.207874	2938.251	1.776343	2940.266	2.782725	2906.67
28	2.339221	3190.212	0.934686	3149.936	0.087078	3178.847	2.453912	3176.882	3.837248	3150.39	2.144217	3040.08	11.18717	2989.437	29.09572	2913.69
29	3.125715	3058.531	0.099618	3096.039	2.836937	3057.644	31164.66	33448.12	3.597712	3036.214	0.569796	3006.121	1.964131	3007.683	0.163264	3000.817
30	3.692203	3158.388	0.228616	3182.424	3.597296	3155.898	156730.4	75402.45	3.316564	3134.025	0.396185	3104.812	3.269537	3110.979	0.210976	3100.568
win	24	30	17	26	18	30	25	30	24	30	22	25	21	28		