An Adaptive Framework to Tune the Coordinate Systems in Nature-Inspired Optimization Algorithms

Zhi-Zhong Liu, Yong Wang, *Senior Member, IEEE*, Shengxiang Yang, *Senior Member, IEEE*, and Ke Tang, *Senior Member, IEEE*

***Abstract*—The performance of many nature-inspired optimization algorithms depends strongly on their implemented coordinate system. However, the commonly used coordinate system is fixed and not well suited for different function landscapes, nature-inspired optimization algorithms thus might not search efficiently. To overcome this shortcoming, in this paper we propose a framework, named ACoS, to $a$daptively tune the coordinate $s$ystems in nature-inspired optimization algorithms. In ACoS, an Eigen coordinate system is established by making use of the cumulative population distribution information, which can be obtained based on a covariance matrix adaptation strategy and an additional archiving mechanism. Since the population distribution information can reflect the features of the function landscape to some extent, nature-inspired optimization algorithms in the Eigen coordinate system have the capability to identify the modality of the function landscape. In addition, the Eigen coordinate system is coupled with the original coordinate system, and they are selected according to a probability vector. The probability vector aims to determine the selection ratio of each coordinate system for each individual, and is adaptively updated based on the collected information from the offspring. ACoS has been applied to two of the most popular paradigms of nature-inspired optimization algorithms, i.e., particle swarm optimization and differential evolution, for solving 30 test functions with 30 and 50 dimensions at the 2014 IEEE Congress on Evolutionary Computation. The experimental studies demonstrate its effectiveness.**

***Index Terms*—Nature-inspired optimization algorithms, particle swarm optimization, differential evolution, coordinate system, adaptive framework.**

This work was supported in part by the Innovation-Driven Plan in Central South University under Grant 2018CX010, in part by the National Natural Science Foundation of China under Grants 61673397, 61673331 and 61672478, in part by the EU Horizon 2020 Marie Sklodowska-Curie Individual Fellowships (Project ID: 661327), in part by the Engineering and Physical Sciences Research Council of UK under Grant EP/K001310/1, in part by the Hunan Provincial Natural Science Fund for Distinguished Young Scholars (Grant No. 2016JJ1018), and in part by the Graduate Innovation Fund of Hunan Province of China under Grant CX2017B062. (*Corresponding author: Yong Wang*).

Z.-Z. Liu is with the School of Information Science and Engineering, Central South University, Changsha 410083, China (Email: zhizhongliu@csu.edu.cn)

Y. Wang is with the School of Information Science and Engineering, Central South University, Changsha 410083, China, and also with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK. (Email: ywang@csu.edu.cn)

S. Yang is with the Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK, and also with the College of Information Engineering, Xiangtan University, Xiangtan 411105, China. (Email: syang@dmu.ac.uk)

K. Tang is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. (Email: tangk3@sustc.edu.cn)

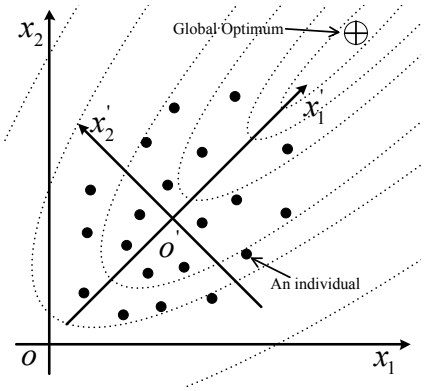


Fig. 1. Advantage of the Eigen coordinate system. In this figure, the dashed ellipses display the contours, ox_1x_2 denotes the original coordinate system, and $o'x_1'x_2'$ refers to the Eigen coordinate system. Compared with ox_1x_2, $o'x_1'x_2'$ is more suitable for the contours since it is established based on the population distribution information.

I. Introduction

Nature-inspired Optimization algorithms (NIOAs) are a class of meta-heuristic algorithms inspired by natural phenomenon. NIOAs usually exploit nature-inspired mechanisms from natural evolution and swarm intelligence to evolve a population of candidate solutions toward the optimal solution. Up to now, numerous NIOA paradigms have been proposed, such as genetic algorithm (GA) [1], evolutionary programming (EP) [2], evolution strategy (ES) [3], genetic programming (GP) [4], differential evolution (DE) [5], particle swarm optimization (PSO) [6], bat algorithm (BA) [7], teaching-learning-based optimization (TLBO) [8], and Jaya algorithm [9]. Compared with other types of optimization methods, NIOAs have some advantages such as ease of use, simple structure, efficiency, and robustness. Therefore, NIOAs have been broadly applied to diverse fields such as renewable energy [10], automotive design [11], route planning [12], classification [13], image processing [14], and action recognition [15].

For many NIOAs, their performance relies crucially on their implemented coordinate system. However, the original coordinate system, which is the most frequently used coordinate system in current NIOAs, is fixed throughout the search process. Under this condition, NIOAs may fail to produce promising solutions matching the requirements of different function landscapes or even one function landscape at different search stages. As a result, it is difficult for NIOAs to search

efficiently in the original coordinate system.

To remedy this issue, in some variants of ES and DE, the Eigen coordinate system is established by making use of the population distribution information. As shown in Fig. 1, the population distribution information can reflect the features of the function landscape to a certain degree and the established Eigen coordinate system (i.e., $o^{'}x_1^{'}x_2^{'}$) is more suitable for the contours compared with the original coordinate system (i.e., ox_1x_2). Therefore, it is expected that NIOAs implemented in the Eigen coordinate system possess the capability to identify the modality of the function landscape and search efficiently. In 2001, a famous ES called CMA-ES was proposed by Hansen and Ostermeier [16]. In CMA-ES, an Eigen coordinate system is established by utilizing the cumulative population distribution information (i.e., the current and historical population distribution information). Afterward, the offspring population is sampled from this Eigen coordinate system. Overall, CMA-ES shows very fast convergence speed and is significantly superior to the ordinary ES. Recently, three attempts (i.e., DE/eig [17] , CoBiDE [18], and CPI-DE [19]) have been made to enhance DE's performance by implementing the crossover operator in both the Eigen coordinate system and the original coordinate system. In DE/eig and CoBiDE, only the current population distribution information is extracted to establish the Eigen coordinate system, while like CMA-ES, in CPI-DE the cumulative population distribution information is employed to construct the Eigen coordinate system. It is interesting to note that all these three attempts in DE draw the similar conclusions: 1) each coordinate system has its own advantages and is suitable for certain kinds of optimization problems, and 2) combining these two coordinate systems can obtain better performance than just using one of them during the evolution. The above conclusions motivate us to design an adaptive scheme to make full use of these two coordinate systems.

This paper presents an adaptive framework, called ACoS, to tune the coordinate systems in NIOAs. ACoS takes advantage of a covariance matrix adaptation strategy and an additional archiving mechanism to extract cumulative population distribution information, with the aim of establishing the Eigen coordinate system. Moreover, this Eigen coordinate system is synthesized with the original coordinate system, and they are selected based on a probability vector. This probability vector determines the selection ratio of each coordinate system for each individual and is adaptively updated according to the collected information from the offspring. ACoS has been applied to two of the most popular NIOA paradigms: PSO and DE. The effectiveness of ACoS has been validated by comprehensive experiments on 30 test functions with 30 and 50 dimensions at the 2014 IEEE Congress on Evolutionary Computation (IEEE CEC2014) [20].

The main contributions of this paper are summarized as follows:

- This paper provides a new point of view toward how to describe some common nature-inspired operators in the original coordinate system, and also offers a convenient transformation from a nature-inspired operator in the original coordinate system to the corresponding nature-inspired operator in the Eigen coordinate system.
- A simple yet effective approach is proposed to establish the Eigen coordinate system, which consists of two main elements, i.e., a covariance matrix adaptation strategy and an additional archiving mechanism. In comparison with the previous methods, the cumulative population distribution information derived from our approach is more sufficient.
- By using a probability vector, this paper presents an adaptive scheme to select an appropriate coordinate system from the original coordinate system and the Eigen coordinate system for each individual during the evolution.
- The proposed framework (i.e., ACoS) can be readily applied to various NIOAs. In this paper, we have verified that ACoS is able to improve the performance of two of the most popular NIOA paradigms: PSO and DE. To the best of our knowledge, it is the first attempt to improve PSO's performance by adjusting the coordinate systems in an adaptive fashion.

The rest of this paper is organized as follows. Section II briefly introduces PSO and DE. Section III presents the coordinate systems and their related work. The proposed ACoS and its implementation details are presented in Section IV. The experimental results and the performance comparisons are given in Section V. Finally, Section VI concludes this paper.

II. Particle Swarm Optimization (PSO) and Differential Evolution (DE)

PSO and DE have become two of the most popular NIOA paradigms. In this section, we will briefly introduce them.

A. Particle Swarm Optimization (PSO)

PSO [6] is inspired by swarm behavior. It searches with a population (called swarm) of candidate solutions (called particles or individuals). Each particle moves in the search space to seek the global optimum, and its movement is guided by its own personal historical best experience as well as the entire swarm's best experience. Due to ease of use and efficiency, PSO has been successful applied to a variety of real-world optimization problems [21].

PSO contains two core equations: the velocity updating equation and the position updating equation. At generation g, PSO updates the dth dimension of the ith particle's velocity $\overrightarrow{v}_i^g = [v_{i,1}^g, v_{i,2}^g, ..., v_{i,D}^g]^T$ and position $\overrightarrow{x}_i^g = [x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g]^T$ as follows:

$$v_{i,d}^{g+1} = v_{i,d}^g + c_1 r_{1,d}(pbest_{i,d}^g - x_{i,d}^g) + c_2 r_{2,d}(gbest_d^g - x_{i,d}^g) \quad (1)$$

$$x_{i,d}^{g+1} = v_{i,d}^{g+1} + x_{i,d}^g \quad (2)$$

where $i \in \{1, \ldots, NP\}$, $d \in \{1, \ldots, D\}$, NP is the population size, D is the dimension of the search space, $\overrightarrow{pbest}_i^g = [pbest_{i,1}^g, pbest_{i,2}^g, ..., pbest_{i,D}^g]^T$ denotes the ith particle's historical best position, $\overrightarrow{gbest}^g = [gbest_1^g, gbest_2^g, ..., gbest_D^g]^T$ means the entire swarm's best position, c_1 and c_2 are the acceleration parameters, and $r_{1,d}$ and $r_{2,d}$ refer to two uniformly distributed random numbers between 0 and 1.

From Eq.(1) and Eq.(2), it is apparent that PSO works dimension by dimension. Based on the updating of each dimension, the whole velocity and position of a particle are updated as follows:

$$\overrightarrow{v}_i^{g+1} = \overrightarrow{v}_i^{g} + c_1\mathbf{R}_1(\overrightarrow{pbest}_i^{g} - \overrightarrow{x}_i^{g}) + c_2\mathbf{R}_2(\overrightarrow{gbest}^{g} - \overrightarrow{x}_i^{g}) \quad (3)$$

$$\overrightarrow{x}_i^{g+1} = \overrightarrow{v}_i^{g+1} + \overrightarrow{x}_i^{g} \quad (4)$$

where $\mathbf{R}_1 = diag(r_{1,1}, r_{1,2}, \ldots, r_{1,D})$ and $\mathbf{R}_2 = diag(r_{2,1}, r_{2,2}, \ldots, r_{2,D})$.

Since PSO's inception, many researchers have improved its performance in different ways [22] [23]. One way is to control or adjust the particle's velocity. Shi and Eberhart [24] incorporated an inertial weight w into the original PSO's velocity updating, and Eq.(3) is thus modified into Eq.(5)

$$\overrightarrow{v}_i^{g+1} = w\,\overrightarrow{v}_i^{g} + c_1\mathbf{R}_1(\overrightarrow{pbest}_i^{g} - \overrightarrow{x}_i^{g}) + c_2\mathbf{R}_2(\overrightarrow{gbest}^{g} - \overrightarrow{x}_i^{g}) \quad (5)$$

The only difference between Eq.(3) and Eq.(5) is that in Eq.(5) w is attached to $\overrightarrow{v}_i^{g}$. In Eq.(5), the value of w decreases linearly from 0.9 to 0.4 over the course of search. It is because a larger w in the early stage of evolution is beneficial to exploration, and a smaller w in the later stage of evolution can facilitate the exploitation. In addition, Clerc and Kennedy [25] introduced a constriction factor χ to reformulate the particle's velocity updating:

$$\overrightarrow{v}_i^{g+1} = \chi[\overrightarrow{v}_i^{g} + c_1\mathbf{R}_1(\overrightarrow{pbest}_i^{g} - \overrightarrow{x}_i^{g}) + c_2\mathbf{R}_2(\overrightarrow{gbest}^{g} - \overrightarrow{x}_i^{g})] \quad (6)$$

where $\chi = 2/|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|$ and $\varphi = c_1 + c_2$. χ is preferably set to 0.729 together with $c_1 = c_2 = 2.05$. For the sake of convenience, the PSO variants in [24] and [25] are called PSO-w and PSO-cf in this paper, respectively, which are two of the most popular PSO variants.

B. Differential Evolution (DE)

DE [5] is another simple yet efficient NIOA paradigm which has been successfully used to deal with a wide spectrum of optimization problems [26]. Similar to other NIOAs, DE searches with a population of NP individuals: $\mathbf{P}^g = \{\overrightarrow{x}_i^{g} = [x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g]^T, i = 1, 2, ..., NP\}$, where g denotes the generation number, NP means the population size, and D refers to the dimension of the search space. In DE, at generation $g = 0$, the initial population $\mathbf{P}^0$ is randomly sampled from the search space. After initialization, DE employs mutation, crossover, and selection operators to steer the population toward the global optimum.

Mutation: The aim of the mutation operator is to generate a mutant vector $\overrightarrow{v}_i^{g}$ for each individual $\overrightarrow{x}_i^{g}$ (also called a target vector). The following are four commonly used mutation operators in the literature:

- DE/rand/1

$$\overrightarrow{v}_i^{g} = \overrightarrow{x}_{r_1}^{g} + F * (\overrightarrow{x}_{r_2}^{g} - \overrightarrow{x}_{r_3}^{g}) \quad (7)$$

- DE/rand/2

$$\overrightarrow{v}_i^{g} = \overrightarrow{x}_{r_1}^{g} + F * (\overrightarrow{x}_{r_2}^{g} - \overrightarrow{x}_{r_3}^{g}) + F * (\overrightarrow{x}_{r_4}^{g} - \overrightarrow{x}_{r_5}^{g}) \quad (8)$$

- DE/current-to-best/1

$$\overrightarrow{v}_i^{g+} = \overrightarrow{x}_i^{g} + F * (\overrightarrow{x}_{best}^{g} - \overrightarrow{x}_i^{g}) + F * (\overrightarrow{x}_{r_1}^{g} - \overrightarrow{x}_{r_2}^{g}) \quad (9)$$

- DE/rand-to-best/1

$$\overrightarrow{v}_i^{g} = \overrightarrow{x}_{r_1}^{g} + F * (\overrightarrow{x}_{best}^{g} - \overrightarrow{x}_{r_1}^{g}) + F * (\overrightarrow{x}_{r_2}^{g} - \overrightarrow{x}_{r_3}^{g}) \quad (10)$$

where the indices r_1, r_2, r_3, r_4, and r_5 are mutually different integers randomly selected from $[1, 2, ..., NP]$ and are also different from i, $\overrightarrow{x}_{best}^{g}$ denotes the best target vector in the current population, and F refers to the scaling factor.

Crossover: After mutation, the crossover operator is performed on each pair of $\overrightarrow{x}_i^{g}$ and $\overrightarrow{v}_i^{g}$ to generate a trial vector $\overrightarrow{u}_i^{g} = [u_{i,1}^g, u_{i,2}^g, ..., u_{i,D}^g]^T$. The binomial crossover is expressed as follows:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, \text{if } rand_j \le CR \text{ or } j = j_{rand} \\ x_{i,j}^g, \text{otherwise} \end{cases} \quad (11)$$

where j_{rand} is a random integer between 1 and D, $rand_j$ is a uniformly distributed random number between 0 and 1 for each j, and CR denotes the crossover control parameter. The condition "$j = j_{rand}$" makes $\overrightarrow{u}_i^{g}$ different from $\overrightarrow{x}_i^{g}$ by at least one dimension.

From Eq.(11), it is easy to see that the crossover operator is implemented dimension by dimension. The updating of the whole trial vector can be described as follows:

$$\overrightarrow{u}_i^{g} = \overrightarrow{x}_i^{g} + \mathbf{C}_r(\overrightarrow{v}_i^{g} - \overrightarrow{x}_i^{g}) \quad (12)$$

where $\mathbf{C}_r = diag(s_1, s_2, ..., s_D)$, and $s_j = \begin{cases} 1, \text{if } rand_j \le CR \text{ or } j = j_{rand} \\ 0, \text{otherwise} \end{cases}$, $j = 1, 2, ..., D$.

Selection: The selection operator is designed to select the better one between $\overrightarrow{u}_i^{g}$ and $\overrightarrow{x}_i^{g}$ to enter the next generation. For a minimization problem, it can be described as follows:

$$\overrightarrow{x}_i^{g+1} = \begin{cases} \overrightarrow{u}_i^{g}, \text{if } f(\overrightarrow{u}_i^{g}) \le f(\overrightarrow{x}_i^{g}) \\ \overrightarrow{x}_i^{g}, \text{otherwise} \end{cases} \quad (13)$$

DE has also attracted much attention [27] and a considerable number of DE variants have been proposed. Among them, jDE [28], SaDE [29], and JADE [30] are three state-of-the-art DE variants. jDE is a DE with self-adaptive control parameter settings [31]. It encodes the control parameters F and CR into individual level and evolves them. SaDE adaptively adjusts the trial vector generation strategies and control parameter settings simultaneously by learning from the previous experience. It maintains a strategy candidate pool which contains four different trial vector generation strategies. Each individual selects a trial vector generation strategy from the pool in an adaptive way to yield its trial vector. JADE is an adaptive DE with an optional external archive. In JADE, the "DE/current-to-$p$best/1" mutation operator exploits the information of multiple best individuals in the population. Moreover, the optional external archive utilizes the difference between the current solutions and the recently explored inferior solutions to produce promising directions. JADE generates F and CR based on their historical record of success.

III. Coordinate Systems and Their Related Work

A. Coordinate Systems

In this subsection, we will introduce the original coordinate system, the Eigen coordinate system, and the difference between them.

1) Original Coordinate System: The original coordinate system is a default coordinate system in most NIOAs. It is formed by the columns of the unity matrix **I**, and thus is a fixed coordinate system. The nature-inspired operators of PSO and DE introduced in Section II are conducted in the original coordinate system. By analyzing these operators, we find that each of them can be described with the usage of three elements: the coefficients, the diagonal matrixes, and the vectors. Therefore, we propose a new point of view toward how to describe these operators in the original coordinate system:

$$\overrightarrow{r}_O = \sum_{i=1}^{m} \alpha_i \overrightarrow{y}_i + \sum_{i=1}^{n} \mathbf{W}_i \overrightarrow{z}_i \tag{14}$$

where $\overrightarrow{r}_O$ denotes the resultant vector, m and n are nonnegative integers, α_i is a coefficient, $\mathbf{W}_i = diag(w_1, w_2, ..., w_D)$, w_1, w_2, ..., w_D are real numbers, and $\overrightarrow{y}_i$ and $\overrightarrow{z}_i$ mean two vectors in the original coordinate system. Eq.(14) can be considered as a general form of the operators in PSO and DE. For example, if $\overrightarrow{r}_O = \overrightarrow{v}_i^{g+1}$, $m = 1$, $\alpha_i = 1$, $\overrightarrow{y}_1 = \overrightarrow{v}_i^{g}$, $n = 3$, $\mathbf{W}_1 = c_1\mathbf{R}_1$, $\overrightarrow{z}_1 = \overrightarrow{pbest}_i^{g}$, $\mathbf{W}_2 = c_2\mathbf{R}_2$, $\overrightarrow{z}_2 = \overrightarrow{gbest}^{g}$, $\mathbf{W}_3 = -(c_1\mathbf{R}_1 + c_2\mathbf{R}_2)$, and $\overrightarrow{z}_3 = \overrightarrow{x}_i^{g}$, then Eq.(14) is revised to

$$\begin{aligned}\overrightarrow{v}_i^{g+1} &= \overrightarrow{v}_i^{g} + \left[c_1\mathbf{R}_1\overrightarrow{pbest}_i^{g} + c_2\mathbf{R}_2\overrightarrow{gbest}^{g} - (c_1\mathbf{R}_1 + c_2\mathbf{R}_2)\overrightarrow{x}_i^{g}\right] \\ &= \overrightarrow{v}_i^{g} + c_1\mathbf{R}_1(\overrightarrow{pbest}_i^{g} - \overrightarrow{x}_i^{g}) + c_2\mathbf{R}_2(\overrightarrow{gbest}^{g} - \overrightarrow{x}_i^{g})\end{aligned} \tag{15}$$

Clearly, Eq.(15) is equivalent to Eq.(3) and both of them are the velocity updating equation in PSO. Indeed, apart from PSO and DE, Eq.(14) is also an effective way to describe the operators in some other NIOA paradigms such as cultural algorithm [32], artificial bee colony algorithm [33], fireworks algorithm [34], brain storm optimization algorithm [35], and so on.

Note that the right-hand side of Eq.(14) involves two parts: $\sum_{i=1}^{m} \alpha_i \overrightarrow{y}_i$ and $\sum_{i=1}^{n} \mathbf{W}_i \overrightarrow{z}_i$. Since the first part is a linear operation of different vectors, it is irrelevant to the coordinate system. In terms of the second part, $\mathbf{W}_i$ is a diagonal matrix used for scaling $\overrightarrow{z}_i$ within the original coordinate system. Since the original coordinate system is a fixed coordinate system, $\mathbf{W}_i$ can only optimize $\overrightarrow{z}_i$ in the deterministic directions, thus failing to identify the modality of different function landscapes or even a single function landscape at different optimization stages. As a result, the search process guided by Eq.(14) may not be efficient.

Remark 1: Each operator in Section II can be rewritten as Eq.(14). It can be found that the right-hand side of the velocity updating equation in PSO (i.e., Eq.(3)) and the crossover operator in DE (i.e., Eq.(12)) contains the second part (i.e., $\sum_{i=1}^{n} \mathbf{W}_i \overrightarrow{z}_i$), which suggests that these two operators may fail to search efficiently in the original coordinate system.

2) Eigen Coordinate System: In this paper, the Eigen coordinate system is established by the columns of an orthogonal matrix **B**, which comes from the Eigen decomposition of the covariance matrix **C**:

$$\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T \tag{16}$$

where **B** is an orthogonal matrix, $\mathbf{B}^T$ is the transposed matrix of **B**, and **D** is a diagonal matrix. Each column of **B** is an eigenvector of **C**, and each diagonal element of **D** is the square root of an eigenvalue of **C**. The fundamental issue in Eq.(16) is how to construct the covariance matrix **C**. In general, the covariance matrix **C** is constructed and updated according to the feedback information resulting from the search process. Therefore, unlike the original coordinate system, the Eigen coordinate system is dynamic throughout the search process, with the aim of suiting the function landscape.

Next, we will discuss how to construct a nature-inspired operator in the Eigen coordinate system. It contains three steps. Firstly, $\mathbf{B}^T$ is applied to transform the vectors in the original coordinate system into the Eigen coordinate system. Subsequently, these vectors in the Eigen coordinate system are combined with the coefficients and diagonal matrixes following Eq.(14), and thus an offspring vector is obtained. Finally, this offspring vector is transformed back into the original coordinate system by making use of **B**, with the aim of evaluating its fitness. Specifically, a nature-inspired operator in the Eigen coordinate system can be described as:

$$\begin{aligned}\overrightarrow{r}_E &= \mathbf{B}\left(\sum_{i=1}^{m} \alpha_i(\mathbf{B}^T\overrightarrow{y}_i) + \sum_{i=1}^{n} \mathbf{W}_i(\mathbf{B}^T\overrightarrow{z}_i)\right) \\ &= \sum_{i=1}^{m} \alpha_i \overrightarrow{y}_i + \sum_{i=1}^{n} \mathbf{B}\mathbf{W}_i\mathbf{B}^T\overrightarrow{z}_i\end{aligned} \tag{17}$$

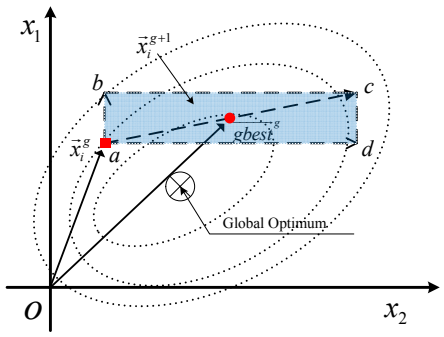
where $\overrightarrow{r}_E$ denotes the resultant vector. By comparing Eq.(17) with Eq.(14), it can be seen that: if we replace $\mathbf{W}_i$ with $\mathbf{B}\mathbf{W}_i\mathbf{B}^T$ on the right-hand side of Eq.(14), then a nature-inspired operator in the original coordinate system is transformed into the corresponding nature-inspired operator in the Eigen coordinate system. Compared with $\mathbf{W}_i\overrightarrow{z}_i$, in $\mathbf{B}\mathbf{W}_i\mathbf{B}^T\overrightarrow{z}_i$, $\mathbf{B}^T$ transforms $\overrightarrow{z}_i$ into the Eigen coordinate system, then $\mathbf{W}_i$ scales $\mathbf{B}^T\overrightarrow{z}_i$ within the Eigen coordinate system, and finally **B** transforms the vector $\mathbf{W}_i\mathbf{B}^T\overrightarrow{z}_i$ back into the original coordinate system.

Remark 2: PSO's velocity updating equation (i.e., Eq.(3)) and DE's crossover operator (i.e., Eq.(12)) in the Eigen coordinate system can be expressed as Eq.(18) and Eq.(19), respectively.

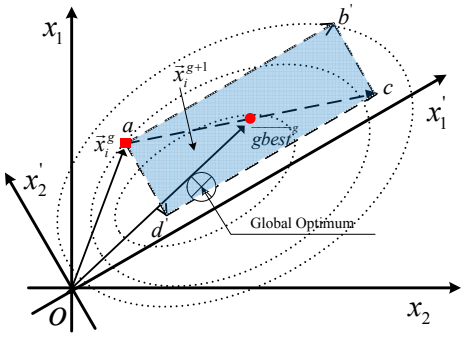
$$\overrightarrow{v}_i^{g+1} = \overrightarrow{v}_i^{g} + c_1\mathbf{B}\mathbf{R}_1\mathbf{B}^T(\overrightarrow{pbest}_i^{g} - \overrightarrow{x}_i^{g}) + c_2\mathbf{B}\mathbf{R}_2\mathbf{B}^T(\overrightarrow{gbest}^{g} - \overrightarrow{x}_i^{g}) \tag{18}$$

$$\overrightarrow{u}_i^{g} = \overrightarrow{x}_i^{g} + \mathbf{B}\mathbf{C}_r\mathbf{B}^T(\overrightarrow{v}_i^{g} - \overrightarrow{x}_i^{g}) \tag{19}$$

3) Difference Between the Original Coordinate System and the Eigen Coordinate System: Next, we will investigate NIOAs' search behaviors in the original and Eigen coordinate systems. To make a clear explanation, we take the basic PSO as an example. For simplicity, suppose that the velocity $\overrightarrow{v}_i^{g}$ of a particle is equal to $\overrightarrow{0}$, $c_1 = c_2 = 2$, the position $\overrightarrow{x}_i^{g}$ is just its historical best position $\overrightarrow{pbest}_i^{g}$, and the dimension of the search space is equal to two. As a result, in the original coordinate system, the new velocity $\overrightarrow{v}_i^{g+1}$ is updated as Eq.(20), and



(a) PSO in the original coordinate system (i.e., ox_1x_2)



(b) PSO in the Eigen coordinate system (i.e., $ox_1'x_2'$)

Fig. 2. PSO works in different coordinate systems. In this figure, the dashed ellipses display the contours, $\overrightarrow{x}_i^g$ denotes the current position, $\overrightarrow{gbest}^g$ means the entire swarm's best position, and $\overrightarrow{x}_i^{g+1}$ is the new position which is located in the blue area.

then the new position $\overrightarrow{x}_i^{g+1}$ is renewed as Eq.(21):

$$\begin{aligned}\overrightarrow{v}_i^{g+1} &= \overrightarrow{v}_i^g + c_1\mathbf{R}_1(\overrightarrow{pbest}_i^g - \overrightarrow{x}_i^g) + c_2\mathbf{R}_2(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g)\\ &= \overrightarrow{0} + c_1\mathbf{R}_1(\overrightarrow{0}) + 2\mathbf{R}_2(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g)\\ &= 2\mathbf{R}_2(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g)\end{aligned} \tag{20}$$

$$\overrightarrow{x}_i^{g+1} = \overrightarrow{x}_i^g + \overrightarrow{v}_i^{g+1} = \overrightarrow{x}_i^g + 2\mathbf{R}_2(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g) \tag{21}$$

where $\mathbf{R}_2 = diag(r_{2,1}, r_{2,2})$, and $r_{2,1}$ and $r_{2,2}$ are two uniformly distributed random numbers between 0 and 1. By replacing $\mathbf{R}_2$ with $\mathbf{BR}_2\mathbf{B}^T$ in Eq.(21), the new position $\overrightarrow{x}_i^{g+1}$ is generated in the Eigen coordinate system:

$$\overrightarrow{x}_i^{g+1} = \overrightarrow{x}_i^g + 2\mathbf{BR}_2\mathbf{B}^T(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g) \tag{22}$$

Fig. 2 shows the difference between PSO in the original coordinate system (Fig. 2(a)) and in the Eigen coordinate system (Fig. 2(b)) for an optimization problem with variable correlation. The original coordinate system is fixed and denoted as ox_1x_2. As pointed out, the Eigen coordinate system is dynamically updated during the search process. Suppose that for this example the Eigen coordinate system is $ox_1'x_2'$, which can suit the contours well. In Fig. 1, $\overrightarrow{x}_i^{g+1}$ in the original coordinate system and the Eigen coordinate system is generated as Eq.(23) and Eq.(24), respectively.

$$\begin{aligned}\overrightarrow{x}_i^{g+1} &= \overrightarrow{x}_i^g + 2\mathbf{R}_2(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g)\\ &= \overrightarrow{x}_i^g + r_{2,1} * \overrightarrow{ab} + r_{2,2} * \overrightarrow{ad}\end{aligned} \tag{23}$$

$$\begin{aligned}\overrightarrow{x}_i^{g+1} &= \overrightarrow{x}_i^g + 2\mathbf{BR}_2\mathbf{B}^T(\overrightarrow{gbest}^g - \overrightarrow{x}_i^g)\\ &= \overrightarrow{x}_i^g + r_{2,1} * \overrightarrow{ab}' + r_{2,2} * \overrightarrow{ad}'\end{aligned} \tag{24}$$

Since $r_{2,1}$ and $r_{2,2}$ are two uniformly distributed random numbers between 0 and 1, $\overrightarrow{x}_i^{g+1}$ generated in the original and Eigen coordinate systems can be any point in the rectangular areas $abcd$ and $ab'cd'$, respectively. As shown in Fig. 2, $abcd$ does not contain the global optimal solution, while $ab'cd'$ contains the global optimal solution and its neighborhood. This phenomenon signifies that PSO may search more efficiently in the Eigen coordinate system.

B. Related Work on the Eigen Coordinate System

In this paper, the related work on the Eigen coordinate system are classified into two categories, according to the way of conducting the nature-inspired operators.

In the first category, the nature-inspired operators are implemented only in the Eigen coordinate system. In 2001, CMA-ES [16] was proposed which samples the offspring population according to:

$$\begin{aligned}\overrightarrow{x}_i^{g+1} &= \overrightarrow{m}^g + \sigma^g\mathcal{N}(\overrightarrow{0}, \mathbf{C}^g),\ i = 1, 2, ..., \lambda\\ &= \overrightarrow{m}^g + \sigma^g(\mathbf{C}^g)^{1/2}\mathcal{N}(\overrightarrow{0}, \mathbf{I}),\ i = 1, 2, ..., \lambda\\ &= \overrightarrow{m}^g + \sigma^g\mathbf{B}^g\mathbf{D}^g(\mathbf{B}^g)^T\mathcal{N}(\overrightarrow{0}, \mathbf{I}),\ i = 1, 2, ..., \lambda\end{aligned} \tag{25}$$

where $\overrightarrow{m}^g$ denotes the mean vector of the search distribution at generation g, σ^g denotes the step size, $\mathbf{C}^g$ refers to a covariance matrix, $\mathbf{B}^g$ is an orthogonal matrix, $\mathbf{D}^g$ is a diagonal matrix, $\mathcal{N}(\overrightarrow{0}, \mathbf{C}^g)$ is a multivariate normal distribution with zero mean and covariance matrix $\mathbf{C}^g$, and $\mathcal{N}(\overrightarrow{0}, \mathbf{I})$ is a multivariate normal distribution with zero mean and identity covariance matrix $\mathbf{I}$. By comparing Eq.(25) with Eq.(17), it can be found that Eq.(25) is a special case of Eq.(17), which means that the sampling operation of CMA-ES only occurs in the Eigen coordinate system. In CMA-ES, this Eigen coordinate system comes from the Eigen decomposition of the covariance matrix $\mathbf{C}^g$, and two strategies, namely the rank-μ-update strategy and the rank-one-update strategy [36], are designed to adapt $\mathbf{C}^g$. In the rank-μ-update strategy, a weighted combination of the μ best out of λ offspring is used to compute $\mathbf{C}_\mu^{g+1}$, which is an estimator of the distribution of the current population:

$$\mathbf{C}_\mu^{g+1} = \sum_{i=1}^{\mu} w_i(\overrightarrow{x}_{i:\lambda}^{g+1} - \overrightarrow{m}^g)(\overrightarrow{x}_{i:\lambda}^{g+1} - \overrightarrow{m}^g)^T \tag{26}$$

where w_i is the ith weight coefficient, λ is the population size, and $\overrightarrow{x}_{i:\lambda}^{(g+1)}$ means the ith best individual among the λ offspring. Thereafter, the information from both the previous and current generations are used to compute the covariance matrix $\mathbf{C}^{g+1}$:

$$\mathbf{C}^{g+1} = (1 - c_\mu)\mathbf{C}^g + \frac{c_\mu}{(\sigma^g)^2}\mathbf{C}_\mu^{g+1} \tag{27}$$

where c_μ is the learning rate for the rank-μ-update strategy. In terms of the rank-one-update strategy, it exploits correlation between consecutive generations and constructs an evolution path to update the covariance matrix. Thus, its implementation is much more complex than the rank-μ-update strategy. These

two strategies are combined together in CMA-ES to update the covariance matrix. Since CMA-ES is able to detect the features of the function landscape, it shows a significant superiority over the ordinary ES. To further expand CMA-ES, an adaptive encoding mechanism called AE_{CMA} [37] is proposed. In AE_{CMA}, a more general approach for covariance matrix adaptation is proposed, which can be applied to ES and estimation of distribution algorithm [38]. Again, in AE_{CMA}, the evolutionary operators are executed only in the Eigen coordinate system.

In the second category, the nature-inspired operators are considered in both the Eigen and original coordinate systems at each generation of NIOAs. For instance, DE/eig [17] and CoBiDE [18] implement the crossover operator of DE in both the Eigen and original coordinate systems in a random manner. As a result, similar to the classical DE, one trial vector is created for one target vector. In DE/eig, all individuals from the current generation are used to compute the covariance matrix:

$$\mathbf{C}^{g+1}=\frac{1}{NP-1}\sum_{i=1}^{NP}(\overrightarrow{x}_i^g-\frac{1}{NP}\sum_{j=1}^{NP}\overrightarrow{x}_j^g)(\overrightarrow{x}_i^g-\frac{1}{NP}\sum_{j=1}^{NP}\overrightarrow{x}_j^g)^T \tag{28}$$

where NP is the population size, and $\overrightarrow{x}_i^g$ and $\overrightarrow{x}_j^g$ mean the ith and jth individuals, respectively. While in CoBiDE, the $NP^{'}$ best out of the individuals from the current population are employed to update the covariance matrix:

$$\begin{aligned}\mathbf{C}^{g+1} = &\frac{1}{NP^{'}-1}\sum_{i=1}^{NP^{'}}(\overrightarrow{x}_{i:NP}^g-\frac{1}{NP^{'}}\sum_{j=1}^{NP^{'}}\overrightarrow{x}_{j:NP}^g)\times \\ &(\overrightarrow{x}_{i:NP}^g-\frac{1}{NP^{'}}\sum_{j=1}^{NP^{'}}\overrightarrow{x}_{j:NP}^g)^T\end{aligned} \tag{29}$$

where $NP^{'}=ps*NP$, $ps\in[0,1]$ is a user-defined parameter, and $\overrightarrow{x}_{i:NP}^g$ and $\overrightarrow{x}_{j:NP}^g$ denote the ith and jth best individuals, respectively. From Eq.(28) and Eq.(29), it can be seen that only the current population distribution information is utilized to compute the covariance matrix. Very recently, a novel DE framework called CPI-DE [19] is proposed. In CPI-DE, DE's crossover operator is executed in both the Eigen and original coordinate systems in a deterministic manner and, consequently, two trial vectors are generated for each target vector. Thereafter, the best one among the target vector and its two trial vectors will survive into the next generation. The covariance matrix in CPI-DE is estimated by the rank-NP-update strategy, which can be regarded as an extension of the rank-μ-update strategy in CMA-ES. This rank-NP-update strategy contains two steps. In the first step, the NP best out of $2*NP$ offspring (note that in CPI-DE, the offspring population consists of $2*NP$ trial vectors) are used to estimate the current population distribution:

$$\mathbf{C}_{NP}^{g+1}=\sum_{i=1}^{NP}w_i(\overrightarrow{x}_{i:2*NP}^{g+1}-\overrightarrow{m}^g)(\overrightarrow{x}_{i:2*NP}^{g+1}-\overrightarrow{m}^g)^T \tag{30}$$

where w_i is the ith weight coefficient and $\overrightarrow{x}_{i:2*NP}^{g+1}$ represents the ith best individual in the offspring population. In the second step, the population distribution information from the current and historical generations are used to adapt the covariance matrix:

$$\mathbf{C}^{g+1}=(1-c_{NP})\mathbf{C}^g+\frac{c_{NP}}{(\sigma^g)^2}\mathbf{C}_{NP}^{g+1} \tag{31}$$

where c_{NP} is the learning rate and σ^g is the step size. It is claimed in CPI-DE [19] that there is no necessary to adapt the step size for DE, since DE has a different search pattern with ES. In fact, σ^g is set to 1 in CPI-DE, which means that the covariance matrix is of equal importance at each generation. It is necessary to note that CPI-DE does not utilize the rank-one-update strategy. The reason is that the rank-one-update strategy plays a less important role when the population size is large, and DE usually maintains a relatively large population compared with ES. Besides, the rank-one-update strategy is much more complex than the rank-μ-update strategy. Therefore, by eliminating the rank-one-update strategy, the adaptation of the covariance matrix in CPI-DE becomes simpler. There is an agreement from the above three attempts: the usage of both the Eigen and original coordinate systems at each generation can reach better performance than the usage of one of them during the whole search process.

Our work in this paper falls into the second category. Moreover, the Eigen and original coordinate systems are adaptively tuned as the search process proceeds.

IV. Proposed Approach

A. ACoS

We continue the work on the coordinate systems and propose a novel framework named ACoS. The motivation of ACoS comes from three aspects:

- A large population can provide more information to estimate the Eigen coordinate system, compared with a small population. However, given the maximum number of fitness evaluations, the increase of the population size will lead to the decrease of the generation number, which might cause incomplete convergence of NIOAs. Consequently, it is necessary to design a mechanism to strike the balance between the accuracy of estimation and the convergence performance.
- As introduced in Section III-B, some researchers have recognized the importance of combining the original coordinate system with the Eigen coordinate system in the design of NIOAs. However, the current methods adjust these two coordinate systems in either a random way or a deterministic way. How to exploit the feedback information from the search process to adaptively tune them has not yet been investigated.
- The coordinate systems play a very important role in the performance of many NIOAs. Note, however, that in current studies the coordinate systems have been applied to enhance the performance of few NIOA paradigms (e.g., ES and DE). It is an interesting topic to boost the research on the coordinate systems to other NIOA paradigms.

ACoS aims at addressing the above three issues. In ACoS, an additional archiving mechanism is designed to maintain the

Algorithm 1 Framework of ACoS

```
Initialize g = 0, P^0 = {x⃗_1^0, x⃗_2^0, ..., x⃗_NP^0}, archive A = ∅, and C^0 = B^0 = I;
Initialize the probability vector p⃗ = (p_1, p_2, ..., p_NP) = (0.5, 0.5, ..., 0.5);
while the termination criterion is not met do
    for i = 1 to NP do
        if rand ≤ p_i then
            Implement the nature-inspired operators in the Eigen coordinate system to generate the ith offspring;
        else
            Implement the nature-inspired operators in the original coordinate system to generate the ith offspring;
        end if
    end for
    Evaluate the offspring population;
    Execute the selection operator of NIOAs to get P^{g+1};
    Update A, C^{g+1}, and B^{g+1} based on Section IV-B;
    Update p⃗ according to Section IV-C;
    g = g + 1;
end while
```

offspring not only in the current generation but also in the past several generations. Therefore, sufficient information can be obtained to estimate an appropriate Eigen coordinate system without adding the population size and reducing the generation number. As a result, ACoS achieves a balance between the accuracy of estimation and the convergence performance. Afterward, the Eigen and original coordinate systems are selected in an adaptive way (rather than a random or deterministic way) according to a probability vector, which is updated based on the collected information from the offspring. ACoS can be readily applied to various NIOAs, and in this paper we consider two of the most popular NIOA paradigms: PSO and DE.

The general framework of ACoS has been given in **Algorithm 1**. In **Algorithm 1**, $rand$ denotes a uniformly distributed random number on the interval $[0, 1]$. In the initialization process, the population $\mathbf{P}^0 = \{\overrightarrow{x}_1^0, \overrightarrow{x}_2^0, ..., \overrightarrow{x}_{NP}^0\}$ is randomly sampled from the search space, the archive $\mathbf{A}$ is initialized to be empty, the covariance matrix $\mathbf{C}^0$ and the orthogonal matrix $\mathbf{B}^0$ are set to be the unity matrix $\mathbf{I}$, and the probability vector $\overrightarrow{p} = (p_1, p_2, ..., p_{NP})$ is initialized to be $\overrightarrow{p} = (0.5, 0.5, ..., 0.5)$. During the search process, to generate the ith offspring, the operators of NIOAs are implemented in the Eigen and original coordinate systems with the probabilities p_i and $1 - p_i$, respectively. Afterward, the offspring population is evaluated and the selection operator of NIOAs is executed to obtain $\mathbf{P}^{g+1}$. Subsequently, $\mathbf{A}$, $\mathbf{C}^{g+1}$, and $\mathbf{B}^{g+1}$ are updated according to Section IV-B. Finally, $\overrightarrow{p}$ is renewed according to Section IV-C.

Obviously, ACoS is different from the canonical NIOAs due to the simultaneous use and adaptive tuning of the Eigen and original coordinate systems. Next, we will introduce two core components of ACoS: the updating of the Eigen coordinate system and the updating of $\overrightarrow{p}$.

B. Updating of the Eigen Coordinate System

The Eigen coordinate system is updated by making use of an additional archiving mechanism and the rank-μ-update strategy [36].

The additional archiving mechanism adopts an external archive $\mathbf{A}$ to store the offspring in both the current generation and the past several generations. It is because the search area may not change dramatically in the continuous several generations of NIOAs, and thus the offspring in the past several generations, other than the offspring in the current generation, can also provide important information to estimate an appropriate Eigen coordinate system. Actually, the implementation of this additional archiving mechanism is very simple. Firstly, $\mathbf{A}$ is initialized to be an empty set. Then at each generation, the newly generated offspring are added into $\mathbf{A}$. If the archive size (called AS) exceeds a certain threshold, say $3 * NP$ in this paper, then the earlier offspring in $\mathbf{A}$ will be removed based on the "first-in-first-out" rule to keep the archive size at $3 * NP$. Note that unlike the main population $\mathbf{P}^g$, $\mathbf{A}$ does not undergo any nature-inspired operators. Therefore, this additional archiving mechanism can obtain sufficient information to estimate the Eigen coordinate system while never affecting the population size and the generation number.

Subsequently, the rank-μ-update strategy extracts the population distribution information from $\mathbf{A}$. The previous research has demonstrated that the rank-μ-update strategy is an efficient technique to adapt the covariance matrix [36]. In this paper, the size of $\mathbf{A}$ is larger than that of $\mathbf{P}^g$. Therefore, the rank-μ-update can benefit from this relatively larger size to get a reliable estimator of the covariance matrix. Before executing the rank-μ-update strategy, we need to initialize the mean vector of the search distribution $\overrightarrow{m}^g$ in Eq.(26) and the covariance matrix $\mathbf{C}^g$. In this paper, $\overrightarrow{m}^0$ is set to be a randomly generated point in the search space and $\mathbf{C}^0$ is set to be the unity matrix $\mathbf{I}$. Then, at generation $g + 1$, $\overrightarrow{m}^{g+1}$ is updated according to:

$$\overrightarrow{m}^{g+1} = \sum_{i=1}^{\mu} \omega_i \overrightarrow{a}_{i:AS} \tag{32}$$

where $\mu = AS/2$ is the number of the selected solutions, $\overrightarrow{a}_{i:AS}$ denotes the ith best solution out of $\mathbf{A}$ (i.e., $f(\overrightarrow{a}_{1:AS}) \leq f(\overrightarrow{a}_{2:AS}) \leq ... \leq f(\overrightarrow{a}_{\mu:AS})$), and ω_i refers to the ith weight coefficient computed as:

$$\omega_i = \frac{\ln(\mu + 0.5) - \ln i}{n \ln(\mu + 0.5) - \sum_{j=1}^{\mu} \ln i}, i = 1, 2, ..., \mu \tag{33}$$

It can be found that the increase of i will lead to the decrease of ω_i, which implies the better individual will play an more important role in the updating of the $\overrightarrow{m}^{g+1}$. Afterward, an estimator of the current population distribution $\mathbf{C}_{\mu}^{g+1}$ is obtained by:

$$\mathbf{C}_{\mu}^{g+1} = \sum_{i=1}^{\mu} \omega_i (\overrightarrow{a}_{i:AS} - \overrightarrow{m}^g)(\overrightarrow{a}_{i:AS} - \overrightarrow{m}^g)^T \tag{34}$$

Algorithm 2 Updating of the probability vector $\overrightarrow{p}$

switch (the case of the collected information from the offspring)
case *Eigen coordinate system is better*:
  $p_i \leftarrow p_i + r(p_i)$;
case *Eigen coordinate system is worse*:
  $p_i \leftarrow p_i - \eta * r(p_i)$;
case *original coordinate system is better*:
  $p_i \leftarrow p_i - r(1 - p_i)$;
case *original coordinate system is worse*:
  $p_i \leftarrow p_i + \eta * r(1 - p_i)$;
end switch

Finally, the covariance matrix $\mathbf{C}^{g+1}$ is updated by making use of the cumulative population distribution information:

$$\mathbf{C}^{g+1} = (1 - c_\mu)\mathbf{C}^g + c_\mu \mathbf{C}_\mu^{g+1} \tag{35}$$

where $c_\mu \approx \frac{1}{3}\mu_{eff}/D^2$ denotes the learning rate, $\mu_{eff} = (\sum_{i=1}^{\mu} \omega_i^2)^{-1}$ is the variance effective selection mass, and D is the dimension of the search space.

After $\mathbf{C}^{g+1}$ is obtained, an Eigen decomposition is performed on $\mathbf{C}^{g+1}$ according to Eq.(16) to produce the orthogonal matrix $\mathbf{B}^{g+1}$, the columns of which form the Eigen coordinate system.

C. Updating of the Probability Vector $\overrightarrow{p}$

The probability vector $\overrightarrow{p} = (p_1, p_2, ..., p_{NP})$ determines the selection ratio of each coordinate system for each individual. To be specific, for the ith ($i \in \{1, \ldots, NP\}$) individual, ACoS implements the nature-inspired operators in the Eigen and original coordinate systems with the probabilities p_i and $1 - p_i$, respectively. Since there is no priori knowledge about the characteristics of the function landscapes, the Eigen and original coordinate systems are considered to be of equal importance at the beginning of search process, i.e., $\overrightarrow{p} = (0.5, 0.5, ..., 0.5)$. Then, $\overrightarrow{p}$ is adaptively updated according to the collected information derived from the offspring.

In this paper, we collect the information including which coordinate system is used to generate the offspring and how about the quality of the generated offspring. It is easy to identify which coordinate system is used to produce the offspring. However, how to measure the quality of the offspring is usually dependent on a specific NIOA. For PSO, if a particle's new position is better than its personal historical best position, then the offspring performs better, otherwise, it performs worse. In terms of DE, if the trial vector outperforms its corresponding target vector, then the offspring performs better; otherwise, it performs worse. Without loss of generality, the collected information derived from the offspring can be categorized into four cases:

- *Eigen coordinate system is better*: the Eigen coordinate system is used to generate the offspring and the offspring performs better;
- *Eigen coordinate system is worse*: the Eigen coordinate system is used to generate the offspring but the offspring performs worse;
- *Original coordinate system is better*: the original coordinate system is used to generate the offspring and the offspring performs better;
- *Original coordinate system is worse*: the original coordinate system is used to generate the offspring but the offspring performs worse.

These four cases have been considered fully in **Algorithm 2** to adaptively update $\overrightarrow{p}$. The main principle behind **Algorithm 2** is the "use it or lose it" rule: if one coordinate system is used to generate the offspring and the offspring performs better, the selection ratio for this coordinate system will increase; otherwise, the selection ratio for this coordinate system will decrease. More specifically, for the ith individual:

- In the case of *Eigen coordinate system is better*, a reward $r(p_i)$ is added to p_i. $r(\cdot)$ denotes a reward function defined as $r(x) = \varepsilon(1-x)e^{-2x},\ x \in [0, 1]$. In this reward function, $\varepsilon = 0.05$ is a constriction factor to clamp the reward value into $[0, 0.05]$, and $(1-x)e^{-2x}$ is a concave function whose value decreases from 1 to 0 when the variable x increases from 0 to 1. As a result, a larger p_i will receive a smaller $r(p_i)$. It is reasonable since a larger p_i means that the Eigen coordinate system already has more potential to be chosen, and a smaller reward would restrain the dramatic increasing of p_i and adapt p_i to a proper value in a more robust way.
- In the case of *Eigen coordinate system is worse*, a punishment $\eta * r(p_i)$ is added to p_i. In $\eta * r(p_i)$, $\eta = 0.1$ is a punishment coefficient. Therefore, $\eta * r(p_i)$ is smaller than $r(p_i)$, which implies that the case *Eigen coordinate system is worse* has less influence on p_i than the case *Eigen coordinate system is better* at one time. The reason is the following. A NIOA usually is a trial-and-error method and the case *Eigen coordinate system is worse* is more likely to happen compared with the case *Eigen coordinate system is better*. Therefore, the more likely occurred case (i.e., *Eigen coordinate system is worse*) should have less influence on p_i than the less likely occurred case (i.e., *Eigen coordinate system is better*) at one time, due to the fact that these two cases' whole effects on p_i should be similar.
- In the case of *original coordinate system is better*, the selection ratio of the original coordinate system will increase and, therefore, p_i will decrease. The reduced value is equal to $r(1 - p_i)$.
- In the case of *original coordinate system is worse*, the selection ratio of the original coordinate system will decrease and p_i thus will increase. The increased value is equal to $\eta * r(1 - p_i)$.

It is clear that in the above four cases, p_i is updated in different ways. By making use of $\overrightarrow{p}$, NIOAs can dynamically select an appropriate coordinate system from the original coordinate system and the Eigen coordinate system for each individual across the search process.

D. Applying ACoS to PSO and DE

ACoS has a simple structure and can be easily applied to various NIOAs. For a specific NIOA, if it is under the frame-

work of ACoS, it will dynamically select one of the Eigen and original coordinate systems according to $\overrightarrow{p}$ to generate the offspring. Since the updating of the Eigen coordinate system and $\overrightarrow{p}$ has been introduced previously, when implementing a specific NIOA under the framework of ACoS, we only need to consider how to generate the offspring in different coordinate systems and how to use the selection operator. In this paper, we apply ACoS to two of the most popular NIOA paradigms, namely PSO and DE.

For PSO, the offspring are generated via the velocity updating equation and the position updating equation. These two equations in the original coordinate system have been given in Eq.(3) and Eq.(4), respectively. According to Section III, Eq.(4) is irrelevant to the coordinate systems. With respect to Eq.(3), it depends on the coordinate systems and its implementation in the Eigen coordinate system has been given in Eq.(18). It is necessary to note that PSO does not employ the selection operator and, therefore, the selection operator in Step 12 of **Algorithm 1** can be eliminated.

For DE, the offspring are produced through the mutation and crossover operators. These two operators in the original coordinate system have been given in Eqs.(7)-(10) and Eq.(11), respectively. In fact, the mutation operator is independent of the coordinate systems, while the crossover operator relies on the coordinate systems, the implementation of which in the Eigen coordinate system has been given in Eq.(19). In addition, the selection operator of ACoS is the same with that of the original DE.

E. Difference between ACoS and Other Methods

Next, we compare ACoS with other related work introduced in Section III-B. Compared with CMA-ES which samples all the individuals in the Eigen coordinate system, ACoS has some advantages listed as follows:

- It makes use of both the Eigen and original coordinate systems. The Eigen coordinate system enables NIOAs to identify the modality of the fitness landscape and enhance the search efficiency, while the original coordinate system can maintain the superiority of the original NIOAs.
- The updating of the Eigen coordinate system in ACoS is simpler. ACoS eliminates the much more complex rank-one-update strategy and only adopts the rank-μ-update strategy to estimate the Eigen coordinate system. In addition, an additional archiving mechanism with negligible computational cost is designed to improve the estimation accuracy.
- ACoS can be readily applied to many other NIOAs. This can be attributed to the fact that the step-size control, which plays a very important role in CMA-ES, can be ignored in many other NIOAs due to their different search patterns with CMA-ES.

Compared with DE/eig, CoBiDE and CPI-DE which focus on enhancing DE's performance, ACoS has the following advantages:

- ACoS is designed to improve the performance of not only DE but also many other NIOAs.
- To update the Eigen coordinate system, DE/eig and CoBiDE only utilize the current population distribution information, therefore the established Eigen coordinate system might be inappropriate due to insufficient information. In CPI-DE and ACoS, the cumulative population distribution information is used to update the Eigen coordinate system. Note, however, that ACoS employs an additional archiving mechanism which can obtain more sufficient information while having no influence on the population size and the generation number.
- Although both the Eigen and original coordinate systems are utilized in DE/eig, CoBiDE, CPI-DE, and ACoS, DE/eig and CoBiDE adjust these two coordinate systems in a random manner which ignores the feedback information from the search process and, therefore, is not well suited for different kinds of fitness landscapes. In addition, CPI-DE generates two offspring for each target vector, one in the Eigen coordinate system and the other in the original coordinate system, which inevitably spends more fitness evaluations at each generation. In contrast, ACoS adapts these two coordinate systems in an adaptive way as the search process proceeds, thus exploiting the feedback information and producing only one offspring for each target vector simultaneously.

V. Experimental Study

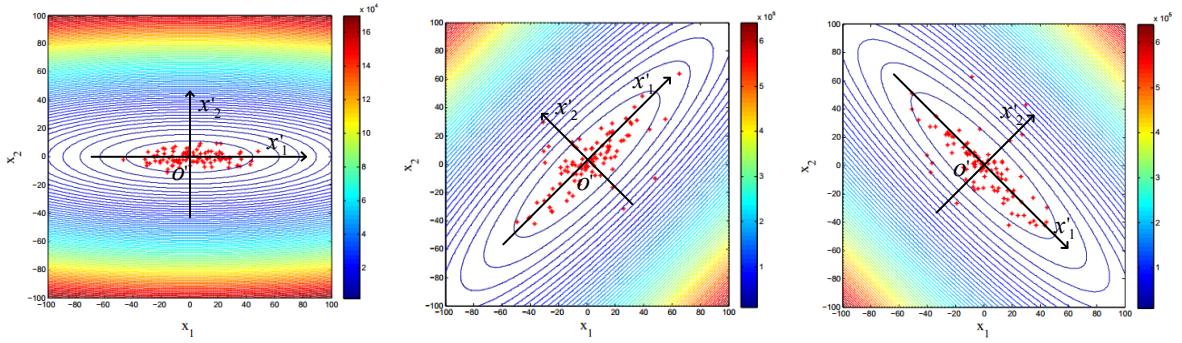
In this section, our experiments were conducted on 30 test functions with 30 dimensions (30D) and 50 dimensions (50D) at IEEE CEC2014. These 30 test functions are denoted as cf_1-cf_{30}, and their details can be available from [20]. In general, these 30 test functions can be grouped into four classes: 1) Unimodal functions cf_1-cf_3; 2) Simple multimodal functions cf_4-cf_{16}; 3) Hybrid functions cf_{17}-cf_{22}; and 4) Composition functions cf_{23}-cf_{30}.

In our experiments, we performed 51 independent runs for each algorithm on each test function. All the experiments were implemented on a PC with Intel Core i5-4590 CPU @ 3.30 GHz and 64-bit Windows 10 operating system. A run will terminate if the maximum number of fitness evaluations (FEs) is reached, which was recommended to be $10000 * D$ [20]. At the end of a run, the function error value $(f(\overrightarrow{x}^{best}) - f(\overrightarrow{x}^*))$ was recorded, where $\overrightarrow{x}^*$ is the optimal solution and $\overrightarrow{x}^{best}$ denotes the best solution found. If the function error value is less than 10^{-8}, it was taken as zero. The average and standard deviation of the function error values in all runs (denoted as "Mean Error" and "Std Dev") were used to measure the performance of an algorithm. Besides, to test the statistical significance of the experimental results between two algorithms, the Wilcoxon's rank sum test at a 0.05 significance level was performed.

For the sake of convenience, if a specific NIOA is under the framework of ACoS, the name of this NIOA will be modified by adding four letters "ACoS-". For example, PSO-w under our framework is named as ACoS-PSO-w.

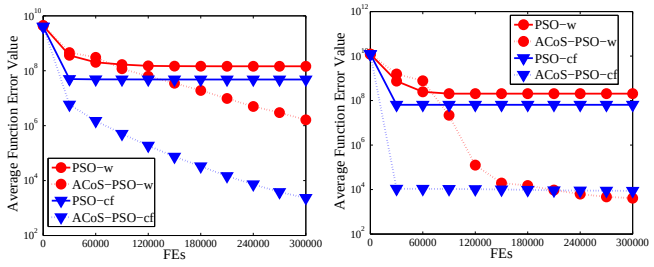
A. Principle Analysis

Firstly, we intend to verify whether ACoS can detect the modality of a function or not. To answer this question, we



(a) Nonrotated ellipsoidal function (b) Rotated ellipsoidal function (rotate 45°) (c) Rotated ellipsoidal function (rotate -45°)

Fig. 3. Contours of three ellipsoidal functions, the trial vectors (marked as red asterisks) of ACoS-DE/rand/1/bin, and the Eigen coordinate systems (denoted as $o^{'}x_1^{'}x_2^{'}$) after 15 generations.



(a) cf_1 with 30D (b) cf_{18} with 30D

Fig. 4. Evolution of the average function error values derived from two popular PSO versions (PSO-w and PSO-cf) and their augmented algorithms versus the number of FEs on cf_1 with 30D and cf_{18} with 30D

took ACoS-DE/rand/1/bin as an example, and tested it on three ellipsoidal functions with two dimensions. Among these three ellipsoidal functions, one is nonrotated problem and the other two are rotated problems which rotate $45°$ and $-45°$, respectively. Therefore, these three ellipsoidal functions can provide different function landscapes for testing. For the parameters of ACoS-DE/rand/1/bin, the population size was set to 100, F was set to 0.9, and CR was set to 0.5. Afterward, we run ACoS-DE/rand/1/bin with 15 generations and calculated the orthogonal matrix $\mathbf{B}$ via the Eigen decomposition in Eq.(16). Fig. 3 depicts the contours of these three ellipsoidal functions, the trial vectors (marked as red asterisks), and the Eigen coordinate systems (denoted as $o^{'}x_1^{'}x_2^{'}$) obtained by making use of $\mathbf{B}$ at the 15th generation.

From Fig. 3, we can observe that the coordinate axes of the Eigen coordinate system rotate about $0°$, $45°$, and $-45°$ compared with the coordinate axes of the original coordinate system for nonrotated ellipsoidal function, rotated ellipsoidal function with $45°$, and rotated ellipsoidal function with $-45°$, respectively. This phenomenon indicates that the Eigen coordinate system in ACoS can be adapted to suit different fitness landscapes. As for the generated trial vectors, they spread out along the valleys, and their distributions are in accordance with the three different function landscapes. Therefore, it can be concluded that ACoS has the capability to identify the modality of a function at hand.

B. ACoS for Two Popular PSO Variants

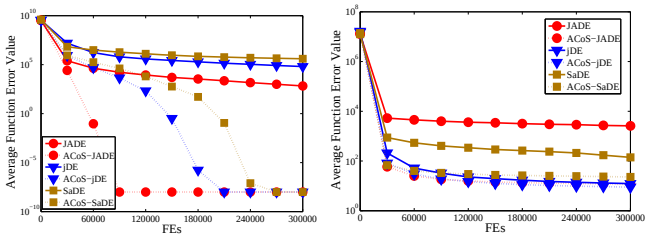
Subsequently, we applied ACoS to two of the most popular PSO variants: PSO-w and PSO-cf, which have been introduced in Section II-A. The resultant methods are denoted as ACoS-PSO-w and ACoS-PSO-cf, respectively.

The population size of these two PSO variants and their augmented algorithms was set to be 40 and 60 when the dimension of the search space was equal to 30 and 50, respectively. The experimental results on cf_1-cf_{30} with 30D and 50D are given in Table S-I and Table S-II in the supplementary file, where "+", "−", and "≈" denote that PSO-w or PSO-cf performs better than, worse than, and similar to its augmented algorithm, respectively. The last three rows of Tables S-I and S-II summarize the experimental results.

Important observations can be obtained from Tables S-I and S-II:

- In the case of $D = 30$, ACoS-PSO-w and ACoS-PSO-cf have an edge over their original algorithms on 27 and 24 test functions, respectively. With respect to $D = 50$, both ACoS-PSO-w and ACoS-PSO-cf achieve better performance than their original algorithms on 26 test functions. However, PSO-w and PSO-cf cannot surpass their augmented algorithms on more than three test functions when $D = 30$ and $D = 50$.
- ACoS-PSO-w and ACoS-PSO-cf are never inferior to their original algorithms on any unimodal functions, hybrid functions, and composition functions, regardless of the number of the decision variables.
- ACoS is able to achieve great performance improvement toward PSO-w and PSO-cf on all the unimodal functions (i.e., cf_1-cf_3), five simple modal functions (i.e., cf_4, cf_7, and cf_{13}-cf_{15}), four hybrid functions (i.e., cf_{17}, cf_{18}, cf_{20}, and cf_{21}), and two composition functions (i.e., cf_{29} and cf_{30}). Moreover, ACoS offers the optimal solutions for three cases in all runs, which have been highlighted in **boldface** in Tables S-I and S-II.
- It seems that the increase of the dimension (i.e., from $D = 30$ to $D = 50$) does not have a remarkable influence on the performance improvement of our framework.

From the above observations, our framework significantly improves the performance of these two popular PSO variants,



(a) cf_1 with 30D (b) cf_{20} with 30D

Fig. 5. Evolution of the average function error values derived from three state-of-the-art DE variants (JADE, jDE, and SaDE) and their augmented algorithms versus the number of FEs on cf_1 with 30D and cf_{20} with 30D

which indicates that: 1) there is a necessity to consider both the Eigen and original coordinate systems in the design of PSO variants, and 2) the adaptive scheme in ACoS is capable of effectively utilizing these two coordinate systems. The convergence graphs of the average function error values derived from these two PSO variants and their augmented algorithms are plotted in Fig. 4 for two test functions (i.e., cf_1 with 30D and cf_{18} with 30D).

C. ACoS for Three State-of-the-Art DE Variants

Thereafter, we investigated the influence of ACoS on three famous DE variants: JADE, jDE, and SaDE, which have been introduced in Section II-B. To ensure the comparison fair, the parameter settings of JADE, jDE, and SaDE were identical with their original papers, and remained unchanged when they were under the framework of ACoS. Table S-III and Table S-IV in the supplementary file show the comparison results on cf_1-cf_{30} with 30D and 50D, where "+", "−", and"≈" denote that a state-of-the-art DE variant performs better than, worse than, and similar to its augmented algorithm, respectively. The last three rows of Tables S-III and S-IV summarize the experimental results.

As can be seen from Tables S-III and S-IV, ACoS significantly improves JADE, jDE, and SaDE on many test functions. Specifically, compared with their original algorithms, when $D = 30$, ACoS-JADE, ACoS-jDE and ACoS-SaDE obtain significance on 17, 13, and 21 test functions, respectively; meanwhile in the case of $D = 50$, they outperform on 12, 14, and 23 test functions, respectively. In contrast, JADE, jDE, and SaDE cannot beat their augmented algorithms on more than four test functions. Besides, under our framework, these three state-of-the-art DE variants can consistently solve 13 cases, which have been highlighted in **boldface** in Tables S-III and S-IV. Moreover, the superiority of ACoS-jDE and ACoS-SaDE over their original algorithms increases as the dimension of the search space increases (i.e., from $D = 30$ to $D = 50$).

The above comparison demonstrates that ACoS can effectively improve the performance of these three state-of-the-art DE variants, which verifies the necessity to consider both the Eigen and original coordinate systems in an adaptive fashion when designing DE variants. Two convergence graphs are given in Fig. 5 for the performance comparison between these three state-of-the-art DE variants and their augmented algorithms.

D. Comparison between ACoS and Other Eigen Coordinate System-Based Methods

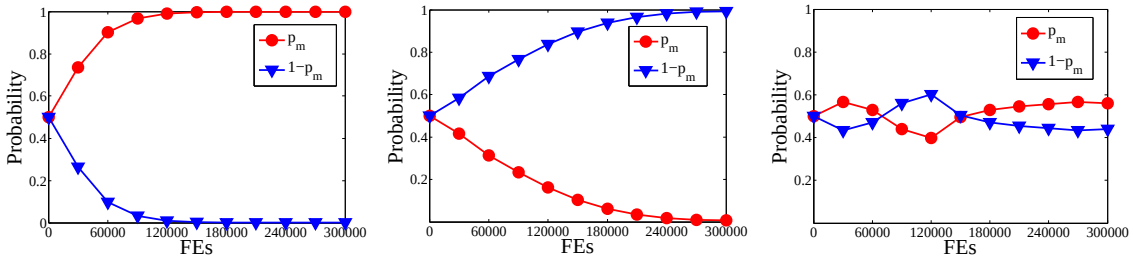
The aim of this subsection is to compare ACoS with other Eigen coordinate system-based methods: CMA-ES, CoBiDE, DE/eig, and CPI-DE, which have been introduced in Section III-B. Due to its outstanding performance, JADE was selected as the instance algorithm. It should be noted that the updating of the Eigen coordinate system in CMA-ES is particularly designed for ES and cannot be used to JADE. With respect to ACoS, CoBiDE, DE/eig, and CPI-DE, we applied them to JADE and obtained ACoS-JADE, CoJADE, JADE/eig, and CPI-JADE, respectively. For fair comparison, ACoS-JADE, CoJADE, JADE/eig, and CPI-JADE adopted the same parameter settings of F, CR, and NP with the original JADE, while the other parameter settings were identical with their own original papers. cf_1-cf_{30} with 30D were employed in the comparative study, and Table S-V summarizes the experimental results, where"+", "−", and "≈" denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of ACoS-JADE, respectively.

As shown in Table S-V, ACoS-JADE exhibits the best performance among the five compared methods. It outperforms CMA-ES, CoJADE, JADE/eig, and CPI-JADE on 22, 14, 16 and 10 test functions, respectively; while only loses on no more than two test functions. It is worth noting that ACoS-JADE is never inferior to the four competitors on any unimodal functions, hybrid functions, and composition functions. Compared with CoJADE and JADE/eig, CPI-JADE and ACoS-JADE reach better performance, which demonstrates the potential of utilizing the cumulative population distribution information rather than the single population distribution information to estimate the Eigen coordinate system. Compared with CPI-JADE, ACoS-JADE's superior performance is largely attributed to the usage of the additional archiving mechanism and the probability vector $\overrightarrow{p}$.

E. Benefit of ACoS's Components

We are interested in identifying the benefit of two crucial components of ACoS: the additional archiving mechanism and the probability vector $\overrightarrow{p}$. To this end, we still selected JADE as the instance algorithm and two groups of experiments were carried out. In the first group, the archiving mechanism was eliminated and the offspring in the current generation played the role of the archive $\mathbf{A}$ in ACoS accordingly, while the other parts were kept untouched. This compared method is denoted as nonAr-ACoS-JADE. With respect to the second group, instead of adaptive tuning, $\overrightarrow{p}$ was fixed during the evolution. We tested three different values for each element of $\overrightarrow{p}$: 0, 0.5, and 1, and these three values represent different conditions, i.e., only the original coordinate system is used, the Eigen and original coordinate systems have an equal probability to be selected, and only the Eigen coordinate system is utilized, respectively. It is evident that the first condition is equivalent to the original JADE. These compared methods are named as JADE, half-ACoS-JADE, and Eig-ACoS-JADE, respectively.

We conducted the experiments on cf_1-cf_{30} with 30D. Experimental results are presented in Table S-VI of the sup-



(a) cf_1 with 30D (b) cf_{10} with 30D (c) cf_{23} with 30D

Fig. 6. Evolution of the average values of p_m and $(1-p_m)$ in ACoS-JADE during the optimization of cf_1 with 30D, cf_{10} with 30D, and cf_{23} with 30D

plementary file, where "+", "−", and "≈" denote that the performance of the corresponding algorithm is better than, worse than, and similar to that of ACoS-JADE, respectively. From Table S-VI, ACoS-JADE performs the best among the five compared methods. Compared with nonAr-ACoS-JADE, ACoS-JADE is significantly better on 11 test functions and does not lose on any test functions. Although ACoS-JADE and nonAr-ACoS-JADE achieve comparable performance on the unimodal functions, ACoS-JADE outperforms nonAr-ACoS-JADE on more complex functions (i.e., simple multimodal functions, hybrid functions, and composition functions). The reason is probably that the additional archiving mechanism preserves the offspring not only in the current generation but also in the past several generations, thus providing sufficient information to estimate a more reliable Eigen coordinate system in complex environments. Compared with JADE, half-ACoS-JADE, and Eig-ACoS-JADE, ACoS-JADE produces better results on 13, 11 and 21 test functions, respectively; while the three competitors cannot outperform ACoS-JADE on more than three test functions. This phenomenon suggests that the updating of $\overrightarrow{p}$ in our framework has the capability to provide a more proper coordinate system. It is noteworthy that ACoS-JADE and half-ACoS-JADE have an advantage over JADE and Eig-ACoS-JADE, which again verifies the effectiveness of combining both the Eigen and original coordinate systems together.

From the above discussion, one can conclude that both the additional archiving mechanism and the probability vector $\overrightarrow{p}$ play very important roles in ACoS. The former is beneficial to estimate a more reliable Eigen coordinate system, and the latter enables each individual to select a more appropriate coordinate system. In addition, the utilization of both the Eigen and original coordinate systems is quite necessary in the design of a NIOA.

F. Evolution of the Probability Vector $\overrightarrow{p}$ in ACoS

Since the probability vector $\overrightarrow{p} = (p_1, p_2, ..., p_{NP})$ determines the selection ratio of each coordinate system for each individual, one may be interested in investigating the dynamic changes of $\overrightarrow{p}$ over the course of search. For this purpose, the mean value of $\overrightarrow{p}$, referred as $p_m = \frac{1}{NP}\sum_{i=1}^{NP} p_i$, is monitored in this subsection.

We still chose JADE as the instance algorithm and tested ACoS-JADE on three test functions with 30D from IEEE CEC2014: the unimodal function cf_1, the simple multimodal function cf_{10}, and the composite function cf_{23}. These three different kinds of test functions aim to provide a comprehensive study on the changes of $\overrightarrow{p}$. To visualize the results, Fig. 6 plots the evolution of the average values of p_m and $(1-p_m)$ over 51 independent runs.

As shown in Fig. 6, there are three different types of curves. In the first type (see Fig. 6(a)), the Eigen coordinate system has a larger probability to be selected than the original coordinate system. Nevertheless, in the second type (see Fig. 6(b)), the situation is opposite. For the third type (see Fig. 6(c)), these two coordinate systems have the similar probability to be chosen over the course of search process. It can be seen from Table S-VI that Eig-ACoS-JADE outperforms JADE on cf_1, JADE surpasses Eig-ACoS-JADE on cf_{10}, and Eig-ACoS-JADE and JADE reach the similar performance on cf_{23}, which implies that the Eigen coordinate system is more appropriate for cf_1, the original coordinate system is a better choice for cf_{10}, and these two coordinate systems are both important for cf_{23}, respectively. Interestingly, the changes of p_m and $(1-p_m)$ in Fig. 6 are consistent with the above analysis, which indicates that ACoS is able to adapt $\overrightarrow{p}$ to a reasonable value to match different function landscapes. In summary, the following concludes can be made: 1) there does not exist a one-size-fits-all coordinate system, and 2) our proposed framework can effectively select the appropriate coordinate system for different optimization problems.

G. Applying ACoS to Other NIOAs

Apart from PSO and DE, we applied ACoS to two other well-known NIOAs: bat algorithm (BA) [7] and teaching-learning-based optimization (TLBO) [15] to validate the generalization of ACoS. According to Sections III and IV, ACoS can be easily applied to BA and TLBO. The comparative experiments were conducted on cf_1-cf_{30} with 30D. To make a fair comparison, the parameter settings of BA and TLBO were identical with their original papers, and kept the same when they were under the framework of ACoS. Table S-VII in the supplementary file provides the experimental results, where "+", "−", and "≈" denote that BA or TLBO performs better than, worse than, and similar to its augmented algorithm, respectively. The last three rows of Table S-IX summarize the experimental results.

From Table S-VII, it is obvious that ACoS has the capability to enhance the performance of both BA and TLBO on a

vast majority of test functions. To be specific, ACoS-BA and ACoS-TLBO outperform their original algorithms on 19 and 25 test functions, respectively. In contrast, BA and TLBO cannot beat their augmented algorithms on any test functions. Hence, we can conclude that ACoS is also an effective framework to improve the performance of these two NIOAs (i.e, BA and TLBO).

Remark 3: In Sections S-I and S-II of the supplementary file, we also analyzed the effect of the parameter settings and the computational time complexity of ACoS, respectively.

VI. Conclusion

An adaptive framework for tuning the coordinate systems in NIOAs, referred as ACoS, was proposed in this paper. ACoS provided a simple yet efficient approach to establish the Eigen coordinate system via an additional archiving mechanism and the rank-μ-update strategy. Thereafter, it adopted a probability vector, which was adaptively updated by making use of the collected information from the offspring, to select an appropriate coordinate system from the Eigen and original coordinate systems for each individual. This paper also presented a new point of view toward how to transform a nature-inspired operator in the original coordinate system into the corresponding nature-inspired operator in the Eigen coordinate system. We applied ACoS to two of the most popular NIOA paradigms, namely PSO and DE, for solving 30 test functions with 30D and 50D from IEEE CEC2014. Simulation results demonstrated that ACoS is capable of significantly enhancing the performance of both PSO and DE. Compared with some other Eigen coordinate system-based methods, ACoS exhibited superior performance. We verified the importance of both the Eigen and original coordinate systems in the design of a NIOA, and the effectiveness of the adaptive tuning of them in ACoS. In addition, ACoS was also applied to two other well-known NIOAs (i.e., BA and TLBO) and achieved improved performance. In the future, we will apply ACoS to some multi-method NIOAs [39], [40].

The Matlab source code of ACoS can be downloaded from Y. Wang's homepage: http://www.escience.cn/people/yongwang1/index.html

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT press, 1992.

[2] L. J. Fogel and A. J. Owens, *Artificial Intelligence Through Simulated Evolution*. London John Wiley, 1966.

[3] I. Rechenberg, "Evolutionsstrategien," in *Simulationsmethoden in der Medizin und Biologie*. Springer, 1978, pp. 83–114.

[4] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT press, 1992.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of 1995 IEEE Int. Conf. Neural Networks*, 1995, pp. 1942–1948.

[7] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp. 65–74, 2010.

[8] R. V. Rao, V. J. Savsani, and D. Vakharia, "Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[9] R. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 7, no. 1, pp. 19–34, 2016.

[10] Y. Wang, H. Liu, H. Long, Z. Zhang, and S. Yang, "Differential evolution with a new encoding mechanism for optimizing wind farm layout," *IEEE Transactions on Industrial Informatics*, 2018, DOI: 10.1109/TI-I.2017.2743761.

[11] Y. Wang, B. Xu, G. Sun, and S. Yang, "A two-phase differential evolution for uniform designs in constrained experimental domains," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 665–680, Oct 2017.

[12] R. Shang, K. Dai, L. Jiao, and R. Stolkin, "Improved memetic algorithm based on route distance grouping for multiobjective large scale capacitated arc routing problems," *IEEE Transactions on Cybernetics*, vol. 46, no. 4, pp. 1000–1013, 2016.

[13] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.

[14] L. Shao, R. Yan, X. Li, and Y. Liu, "From heuristic optimization to dictionary learning: A review and comprehensive comparison of image denoising algorithms," *IEEE Transactions on Cybernetics*, vol. 44, no. 7, pp. 1001–1013, 2014.

[15] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 158–170, 2016.

[16] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.

[17] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 31–49, 2015.

[18] Y. Wang, H.-X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Applied Soft Computing*, vol. 18, pp. 232–247, 2014.

[19] Y. Wang, Z.-Z. Liu, J. Li, H.-X. Li, and G. G. Yen, "Utilizing cumulative population distribution information in differential evolution," *Applied Soft Computing*, vol. 48, pp. 329–346, 2016.

[20] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2013.

[21] Y. Wang, J.-J. Huang, N. Zhou, D.-S. Cao, J. Dong, and H.-X. Li, "Incorporating PLS model information into particle swarm optimization for descriptor selection in QSAR/QSPR," *Journal of Chemometrics*, vol. 29, no. 12, pp. 627–636, 2015.

[22] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, 2016.

[23] S. M. Elsayed, R. A. Sarker, and E. Mezura-Montes, "Self-adaptive mix of particle swarm methodologies for constrained optimization," *Information Sciences*, vol. 277, pp. 216–233, 2014.

[24] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evolutionary Computation*. IEEE, 1998, pp. 69–73.

[25] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[26] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[27] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems." *IEEE Trans. Evolutionary Computation*, vol. 18, no. 5, pp. 689–707, 2014.

[28] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[29] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[30] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[31] Z. Bingul, "Adaptive genetic algorithms applied to dynamic multiobjective problems," *Applied Soft Computing*, vol. 7, no. 3, pp. 791–799, 2007.
[32] R. G. Reynolds, "An introduction to cultural algorithms," in *Proceedings of the third annual conference on evolutionary programming*, vol. 131139. Singapore, 1994.
[33] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.
[34] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *International Conference in Swarm Intelligence*. Springer, 2010, pp. 355–364.
[35] Y. Shi, "Brain storm optimization algorithm," in *International Conference in Swarm Intelligence*. Springer, 2011, pp. 303–309.
[36] N. Hansen, "The CMA evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.
[37] ——, "Variable metrics in evolutionary computation," 2009, https://www.lri.fr/~hansen/hansen-habil-manu.pdf.
[38] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.
[39] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Adaptive configuration of evolutionary algorithms for constrained optimization," *Applied Mathematics and Computation*, vol. 222, pp. 680–711, 2013.
[40] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-based algorithm portfolios for numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782–800, 2010.

Zhi-Zhong Liu received the B.S. degree in automation from Central South University, Changsha, China, in 2013, where he is currently pursuing the Ph.D. degree in control science and engineering.

His current research interests include evolutionary computation, bioinformatics, swarm intelligence, nonlinear equation systems, and multimodal optimization.

Yong Wang (M'08–SM'17) received the B.S. degree in automation from the Wuhan Institute of Technology, Wuhan, China, in 2003, and the M.S. degree in pattern recognition and intelligent systems and the Ph.D. degree in control science and engineering both from the Central South University (CSU), Changsha, China, in 2006 and 2011, respectively.

He is currently an Associate Professor with the School of Information Science and Engineering, CSU. His current research interests include the theory, algorithm design, and interdisciplinary applications of computational intelligence.

Dr. Wang was awarded the Hong Kong Scholar by the Mainland-Hong Kong Joint Postdoctoral Fellows Program, China, in 2013, the Excellent Doctoral Dissertation by Hunan Province, China, in 2013, the New Century Excellent Talents in University by the Ministry of Education, China, in 2013, the 2015 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award, the Hunan Provincial Natural Science Fund for Distinguished Young Scholars, in 2016, the EU Horizon 2020 Marie Sklodowska-Curie Fellowship, in 2016, and a Highly Cited Researcher in computer science by Clarivate Analytics, in 2017. He is currently serving as an associate editor for the *Swarm and Evolutionary Computation*.

Shengxiang Yang (M'00–SM'14) received the Ph.D. degree in systems engineering from Northeastern University, Shenyang, China in 1999.

He is currently a Professor in Computational Intelligence and Director of the Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, U.K. He has over 240 publications.

His current research interests include evolutionary computation, swarm intelligence, computational intelligence in dynamic and uncertain environments, artificial neural networks for scheduling, and relevant real-world applications. He serves as an Associate Editor/Editorial Board Member of seven international journals, such as the *IEEE Transactions on Cybernetics*, *Information Sciences*, *Evolutionary Computation*, and *Soft Computing*.

Ke Tang (M'07–SM'13) received the B.Eng. degree from Huazhong University of Science and Technology, Wuhan, China, in 2002, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2007, respectively.

From 2007 to 2017, he was with the School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, China, first as an Associate Professor (2007-2011) and later a Professor (2011-2017). Currently, he is a Professor at the Department of Computer Science and Engineering, Southern University of Science and Technology (SUSTech), Shenzhen, China. His major research interests include evolutionary computation, machine learning, and their applications.

Dr. Tang has published more than 60 journal papers and more than 70 conference papers. According to Google Scholar, his publications have received more than 4900 citations and the H-index is 32 (as of October 5, 2017). He is an Associate Editor of the *IEEE Transactions on Evolutionary Computation* and served as a member of Editorial Boards for a few other journals. He received the Royal Society Newton Advanced Fellowship (2015) and the 2018 IEEE Computational Intelligence Society Outstanding Early Career Award.

Supplementary File for "An Adaptive Framework to Tune the Coordinate Systems in Nature-Inspired Optimization Algorithms"

Zhi-Zhong Liu, Yong Wang, *Senior Member, IEEE*, Shengxiang Yang, *Senior Member, IEEE*, and Ke Tang, *Senior Member, IEEE*

List of Tables

List of Figures

Z.-Z. Liu is with the School of Information Science and Engineering, Central South University, Changsha 410083, China (Email: zhizhongliu@csu.edu.cn)

Y. Wang is with the School of Information Science and Engineering, Central South University, Changsha 410083, China, and also with the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK. (Email: ywang@csu.edu.cn)

S. Yang is with the Centre for Computational Intelligence (CCI), School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, UK, and also with the College of Information Engineering, Xiangtan University, Xiangtan 411105, China. (Email: syang@dmu.ac.uk)

K. Tang is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. (Email: ketang@ustc.edu.cn)

S-I. Effect of the Parameter Settings in ACoS

ACoS introduces three parameters: the archive size (AS) in Section IV-B, the constriction factor (ε) in Section IV-C, and the punishment coefficient (η) in Section IV-C. The purpose of this subsection is to investigate the sensitivity of these three parameters. To this end, we tested ACoS-JADE on three different kinds of test functions with 30D: the simple multimodal function cf_{16}, the hybrid functions cf_{20}, and the composition function cf_{30}, with the aim of providing multi-facet insights.

Firstly, we investigated how AS affects the performance of ACoS-JADE by testing five different AS values: NP, $2*NP$, $3*NP$, $4*NP$, and $5*NP$. As shown in Fig. S-1 of the supplementary file, AS in the range of $[2*NP, 4*NP]$ is a good choice and $AS = 3*NP$ is highly recommended. The reason is the following. A small AS value (e.g., NP) might not provide enough population distribution information to estimate an appropriate Eigen coordinate system, while a big AS value (e.g., $5*NP$) would contain some outdated population distribution information, thus also having a side effect on the establishment of the Eigen coordinate system.

Subsequently, we studied the sensitivity of the other two parameters: ε and η. As introduced in Section IV-C, ε is used to control the changing scope of p_i at each generation. A small ε value always results in a little change at each generation; thus it will take a lot of generations to reach an appropriate value of p_i. On the contrary, a big ε value will lead to a great oscillation of p_i which causes unstable performance. In essence, NIOAs are randomized methods. Generally, the case that a coordinate system achieves good performance is less likely to occur than the case that a coordinate system achieves worse performance for each individual at each generation. η is employed to alleviate the influence of the more likely occurred case, with the aim of balancing the whole effect of these two cases during the evolution. Neither a small η value nor a big η value can accomplish this goal. Hence, an appropriate value should be set to both ε and η. We chose five different ε values: 0.01, 0.03, 0.05, 0.07, and 0.09, and five different η values: 0.01, 0.05, 0.1, 0.2 and 0.5, and tested the performance of ACoS-JADE with different combinations of ε and η. From Fig. S-2 of the supplementary file, we can observe that, overall, ACoS-JADE exhibits better performance with $\varepsilon \in [0.03, 0.07]$ and $\eta \in [0.05, 0.2]$.

S-II. Analysis of the Computational Time Complexity

When applying ACOS to a NIOA, it will cause extra computational time. It is because ACoS has to update the Eigen coordinate system and the probability vector $\overrightarrow{p}$. From the introduction in Section IV-C, we know that the computational time complexity of the updating of $\overrightarrow{p}$ is very low and is $O(NP)$. In terms of the updating of the Eigen coordinate system, the computational time complexity depends mainly on the calculation of the covariance matrix (i.e, Eq. 27) and the Eigen decomposition (i.e., Eq. 16) [17]. Note that a covariance matrix contains $D \times D$ elements. From Eq. 27, we can infer that the computational time complexity of the calculation of the covariance matrix is $O(D^2 \times AS/2)$. For the Eigen decomposition of a covariance matrix, the Jacobi's method is adopted in this paper, whose computational time complexity is $O(D^3)$. Therefore, the extra computational time complexity of ACoS is $O(D^2 \times max\{D, AS/2\})$.

To ascertain the actual runtime of a NIOA with and without ACoS, Tables S-VIII and S-IX in the supplementary file record the runtime of PSO-w and ACoS-PSO-w, PSO-cf and ACoS-PSO-cf, JADE and ACoS-JADE, jDE and ACoS-jDE, and SaDE and ACoS-SaDE on cf_1-cf_{30} with 30D and 50D, respectively. The extra runtime can be obtained by subtracting the runtime of a NIOA without ACoS from the runtime of this NIOA with ACoS. From Tables S-VIII and S-IX, the extra runtime of ACoS is relatively small and is about 2 seconds and 5 seconds in 30D and 50D, respectively. Therefore, ACoS can be regarded as an efficient framework for NIOAs, since it can significantly improve the performance of NIOAs without much extra computational burden.

With respect to the extra runtime, we would like to further give the following comments:

- The extra runtime is mainly related to the dimension of a test function, and is independent of search algorithms and the fitness evaluations. Considering in many real-world applications, the computational time of the fitness evaluations is relatively high, the extra runtime induced by ACoS thus is trivial to some extent.
- For offline optimization, we do not need to have any concerns about the extra runtime of ACoS since it can be neglected. The extra runtime of ACoS may cause inefficiency for online optimization. Fortunately, we can resort to some efficient programming languages (such as C++), some accelerating techniques (such as GPU accelerator), etc.
- In the future, we will also apply more efficient covariance matrix estimation to reduce the extra runtime of ACoS.

TABLE S-I
EXPERIMENTAL RESULTS OF PSO-W, ACOS-PSO-W, PSO-CF, AND ACOS-PSO-CF OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 30D FROM IEEE CEC2014 USING 300,000 FES.

Test Functions with 30D from IEEE CEC2014		PSO-w Mean Error±Std Dev	ACoS-PSO-w Mean Error±Std Dev	PSO-cf Mean Error±Std Dev	ACoS-PSO-cf Mean Error±Std Dev
Unimodal Functions	cf_1	1.53E+08±1.34E+08−	1.86E+06±3.60E+06	6.68E+07±7.48E+07−	1.08E+03±2.62E+03
	cf_2	1.67E+10±7.67E+09−	1.00E+03±5.14E+03	**7.77E+09±5.94E+09**−	**0.00E+00±0.00E+00**
	cf_3	4.72E+04±3.37E+04−	1.75E+01±1.23E+02	**1.17E+04±1.59E+04**−	**0.00E+00±0.00E+00**
Simple Multimodal Functions	cf_4	1.28E+03±9.00E+02−	1.00E+02±6.30E+01	8.37E+02±9.65E+02−	1.83E+01±2.88E+01
	cf_5	2.07E+01±1.35E-01≈	2.07E+01±1.22E-01	2.02E+01±2.83E-01+	2.07E+01±1.41E-01
	cf_6	2.07E+01±2.92E+00−	1.97E+01±3.30E+00	2.10E+01±3.35E+00−	1.81E+01±4.17E+00
	cf_7	1.76E+02±7.89E+01−	1.06E+01±7.76E+00	1.05E+02±7.63E+01−	1.11E-02±1.34E-02
	cf_8	9.63E+01±2.67E+01−	7.19E+01±1.82E+01	9.59E+01±3.56E+01−	8.62E+01±2.35E+01
	cf_9	1.50E+02±3.10E+01−	1.19E+02±2.38E+01	1.38E+02±4.02E+01−	9.55E+01±2.65E+01
	cf_{10}	3.36E+03±7.42E+02−	2.69E+03±5.82E+02	2.72E+03±8.03E+02+	3.04E+03±6.17E+02
	cf_{11}	3.65E+03±7.20E+02−	3.45E+03±7.27E+02	3.62E+03±6.73E+02≈	3.63E+03±5.65E+02
	cf_{12}	7.19E-01±5.24E-01+	1.17E+00±6.17E-01	3.26E-01±1.08E-01+	6.81E-01±4.43E-01
	cf_{13}	3.03E+00±1.25E+00−	6.99E-01±9.79E-02	2.39E+00±1.25E+00−	4.32E-01±1.04E-01
	cf_{14}	5.11E+01±2.68E+01−	1.19E+00±2.59E-01	4.19E+01±3.13E+01−	5.77E-01±2.56E-01
	cf_{15}	7.52E+04±2.33E+05−	1.08E+03±3.81E+03	1.03E+04±3.49E+04−	5.24E+00±1.70E+00
	cf_{16}	1.13E+01±6.30E-01≈	1.15E+01±5.00E-01	1.14E+01±6.21E-01≈	1.13E+01±6.22E-01
Hybrid Functions	cf_{17}	4.99E+06±5.88E+06−	8.00E+04±2.34E+05	2.26E+06±4.46E+06−	1.73E+03±3.74E+02
	cf_{18}	2.16E+08±4.65E+08−	5.48E+03±5.47E+03	7.51E+07±2.94E+08−	9.38E+03±8.31E+03
	cf_{19}	6.84E+01±6.64E+01−	2.60E+01±2.66E+01	6.00E+01±5.33E+01−	1.13E+01±2.07E+00
	cf_{20}	1.87E+04±2.46E+04−	5.62E+02±8.51E+02	6.61E+03±1.58E+04−	3.21E+02±1.33E+02
	cf_{21}	8.45E+05±1.06E+06−	2.56E+04±9.06E+04	8.93E+05±3.76E+06−	1.20E+03±7.46E+02
	cf_{22}	6.53E+02±3.21E+02−	4.72E+02±1.97E+02	6.36E+02±2.39E+02−	4.93E+02±1.97E+02
Composition Functions	cf_{23}	4.03E+02±6.51E+01−	3.18E+02±7.06E+00	3.65E+02±4.81E+01−	3.15E+02±4.54E-13
	cf_{24}	2.70E+02±2.30E+01−	2.45E+02±7.89E+00	2.61E+02±2.35E+01−	2.42E+02±7.12E+00
	cf_{25}	2.19E+02±1.05E+00−	2.05E+02±2.49E+00	2.16E+02±8.63E+00−	2.07E+02±5.45E+00
	cf_{26}	1.30E+02±6.78E+01−	1.20E+02±6.50E+01	1.32E+02±6.43E+01−	1.14E+02±5.36E+01
	cf_{27}	1.05E+03±1.81E+02−	9.18E+02±2.17E+02	9.16E+02±2.77E+02≈	9.10E+02±2.07E+02
	cf_{28}	1.93E+03±4.66E+02−	1.34E+03±2.97E+02	1.89E+03±4.67E+02−	1.65E+03±4.43E+02
	cf_{29}	1.70E+07±1.54E+07−	8.49E+06±1.05E+07	1.64E+07±1.21E+07−	8.30E+06±1.14E+07
	cf_{30}	2.06E+05±1.89E+05−	1.13E+04±2.43E+04	1.37E+05±1.12E+05−	3.59E+03±1.73E+03
+		1		3	
−		27		24	
≈		2		3	

TABLE S-II
EXPERIMENTAL RESULTS OF PSO-W, ACOS-PSO-W, PSO-CF, AND ACOS-PSO-CF OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 50D FROM IEEE CEC2014 USING 500,000 FES.

Test Functions with 50D from IEEE CEC2014		PSO-w Mean Error±Std Dev	ACoS-PSO-w Mean Error±Std Dev	PSO-cf Mean Error Std Dev	ACoS-PSO-cf Mean Error±Std Dev
Unimodal Functions	cf_1	5.23E+08±2.85E+08−	6.60E+06±7.39E+06	3.30E+08±2.76E+08−	1.23E+05±8.87E+04
	cf_2	5.62E+10±1.57E+10−	1.48E+08±7.73E+08	3.06E+10±1.31E+10−	1.91E+03±4.55E+03
	cf_3	7.77E+04±3.61E+04−	1.76E+02±5.88E+02	**1.42E+04±1.81E+04**−	**0.00E+00±0.00E+00**
Simple Multimodal Functions	cf_4	6.92E+03±3.34E+03−	1.63E+02±1.75E+02	2.24E+03±1.75E+03−	9.10E+01±3.14E+01
	cf_5	2.10E+01±8.68E-02≈	2.10E+01±8.98E-02	2.02E+01±2.65E-01+	2.10E+01±1.44E-01
	cf_6	4.46E+01±5.13E+00−	4.11E+01±4.91E+00	4.10E+01±4.75E+00−	3.87E+01±5.60E+00
	cf_7	4.89E+02±1.52E+02−	1.14E+01±1.91E+01	2.64E+02±1.27E+02−	4.92E-03±7.13E-03
	cf_8	2.75E+02±3,86E+01−	1.79E+02±3.43E+01	2.23E+02±5.38E+01−	1.75E+02±3.69E+01
	cf_9	3.60E+02±6.32E+01−	2.37E+02±5.10E+01	3.18E+02±7.13E+01−	2.03E+02±5.37E+01
	cf_{10}	6.95E+03±1.04E+03−	5.76E+03±1.02E+03	5.72E+03±1.11E+03+	6.34E+03±8.75E+02
	cf_{11}	7.21E+03±9.63E+02−	7.02E+03±1.17E+03	6.91E+03±9.82E+02−	6.67E+03±9.22E+02
	cf_{12}	8.91E-01±6.81E-01+	1.46E+00±8.14E-01	4.64E-01±1.47E-01+	9.89E-01±6.14E-01
	cf_{13}	4.85E+00±8.24E-01−	8.10E-01±1.06E-01	3.50E+00±1.20E+00−	5.19E-01±9.60E-02
	cf_{14}	1.33E+02±3.27E+01−	1.49E+00±2.53E-01	7.60E+01±3.75E+01−	5.63E-01±2.74E-01
	cf_{15}	4.54E+05±5.86E+05−	1.82E+03±6.01E+03	1.65E+05±3.04E+05−	1.02E+01±2.70E+00
	cf_{16}	2.07E+01±7.28E-01≈	2.08E+01±7.38E-01	2.07E+01±8.18E-01≈	2.07E+01±5.94E-01
Hybrid Functions	cf_{17}	2.89E+07±2.86E+07−	3.29E+05±4.22E+05	1.50E+07±2.04E+07−	3.51E+03±2.73E+03
	cf_{18}	1.42E+09±1.12E+09−	1.95E+03±1.61E+03	1.02E+09±9.81E+08−	4.23E+03±1.89E+03
	cf_{19}	3.53E+02±1.82E+02−	6.41E+01±3.97E+01	2.89E+02±2.52E+02−	2.52E+01±9.93E+00
	cf_{20}	5.37E+04±3.90E+04−	9.46E+02±5.10E+02	7.94E+03±1.40E+04−	6.09E+02±1.62E+02
	cf_{21}	9.21E+06±1.10E+07−	2.59E+05±4.70E+05	5.14E+06±8.56E+06−	2.11E+03±7.27E+02
	cf_{22}	1.57E+03±4.43E+02−	1.10E+03±3.12E+02	1.54E+03±3.85E+02−	1.04E+03±2.97E+02
Composition Functions	cf_{23}	6.62E+02±1.45E+02−	3.49E+02±1.87E+01	5.29E+02±1.16E+02−	3.44E+02±4.69E-13
	cf_{24}	4.04E+02±4.57E+01−	2.99E+02±1.83E+01	3.49E+02±3.44E+01−	2.91E+02±6.15E+00
	cf_{25}	2.52E+02±2.15E+01−	2.13E+02±5.08E+00	2.29E+02±1.33E+01−	2.17E+02±9.18E+00
	cf_{26}	1.86E+02±1.19E+02≈	1.88E+02±1.18E+02	1.87E+02±9.95E+01−	1.32E+02±9.67E+01
	cf_{27}	1.75E+03±1.22E+02−	1.52E+03±1.42E+02	1.62E+03±1.80E+02−	1.51E+03±2.09E+02
	cf_{28}	3.68E+03±8.30E+02−	2.42E+03±5.84E+02	3.73E+03±9.88E+02−	2.93E+03±8.20E+02
	cf_{29}	1.23E+08±5.07E+07−	7.00E+07±4.64E+07	1.25E+08±7.59E+07−	3.88E+07±4.56E+07
	cf_{30}	7.35E+05±6.25E+05−	1.96E+04±1.16E+03	5.77E+05±4.36E+05−	1.43E+04±2.44E+03
+		1		3	
−		26		26	
≈		3		1	

TABLE S-III
EXPERIMENTAL RESULTS OF JADE, ACOS-JADE, JDE, ACOS-JDE, SADE, AND ACOS-SADE OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 30D FROM IEEE CEC2014 USING 300,000 FES.

Test Functions with 30D from IEEE CEC2014		JADE	ACoS-JADE	jDE	ACoS-jDE	SaDE	ACoS-SaDE
		Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev	Mean Error±Std Dev
Unimodal Functions	cf_1	**6.09E+02±1.18E+03**−	**0.00E+00±0.00E+00**	**7.35E+04±6.12E+04**−	**0.00E+00±0.00E+00**	**3.60E+05±2.74E+05**−	**0.00E+00±0.00E+00**
	cf_2	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_3	**9.86E-04±5.95E-03**−	**0.00E+00±0.00E+00**	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	**1.92E+01±5.60E+01**−	**0.00E+00±0.00E+00**
Simple Multimodal Functions	cf_4	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	5.09E+00±1.48E+01−	1.24E+00±8.87E+00	**4.13E+01±3.65E+01**−	**0.00E+00±0.00E+00**
	cf_5	2.03E+01±3.23E-02≈	2.03E+01±6.08E-02	2.03E+01±3.80E-02≈	2.03E+01±4.44E-02	2.05E+01±4.94E-02≈	2.03E+01±3.50E-02
	cf_6	9.15E+00±2.21E+00−	6.11E+00±3.54E+00	3.39E+00±3.97E+00−	1.32E+00±2.31E+00	4.86E+00±2.15E+00−	2.93E+00±3.65E+00
	cf_7	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	**1.12E-02±1.50E-02**−	**0.00E+00±0.00E+00**
	cf_8	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	**5.85E-02±2.36E-01**−	**0.00E+00±0.00E+00**
	cf_9	2.62E+01±4.96E+00−	2.47E+01±3.89E+00	4.40E+01±5.33E+00≈	4.30E+01±5.65E+00	3.72E+01±8.61E+00+	4.58E+01±7.67E+00
	cf_{10}	8.16E-03±1.18E-02−	4.89E-03±9.84E-03	1.22E-03±4.94E-03≈	1.22E-03±4.94E-03	2.82E-01±4.35E-01−	4.32E-02±3.08E-02
	cf_{11}	1.67E+03±2.13E+02≈	1.69E+03±1.88E+02	2.41E+03±3.11E+02≈	2.55E+03±3.02E+02	3.25E+03±5.37E+02−	2.62E+03±2.87E+02
	cf_{12}	2.67E-01±3.57E-02≈	2.80E-01±3.76E-02	4.56E-01±6.46E-02+	4.92E-01±6.51E-02	7.95E-01±9.96E-02−	5.13E-01±6.75E-02
	cf_{13}	2.20E-01±3.25E-02≈	2.21E-01±3.64E-02	3.04E-01±3.54E-02−	2.82E-01±3.55E-02	2.66E-01±4.05E-02+	2.99E-01±4.94E-02
	cf_{14}	2.41E-01±3.18E-02−	2.24E-01±3.04E-02	2.83E-01±2.95E-02−	2.63E-01±3.44E-02	2.35E-01±3.70E-02≈	2.31E-01±2.73E-02
	cf_{15}	3.20E+00±4.55E-01≈	3.16E+00±3.76E-01	5.89E+00±7.23E-01≈	5.88E+00±8.14E-01	4.10E+00±1.40E+00+	6.41E+00±8.18E-01
	cf_{16}	9.30E+00±4.61E-01≈	9.41E+00±4.33E-01	9.85E+00±3.81E-01≈	9.99E+00±2.49E-01	1.10E+01±2.64E-01−	1.01E+01±3.12E-01
Hybrid Functions	cf_{17}	1.91E+04±1.08E+05−	4.11E+02±1.56E+02	1.13E+03±9.03E+02−	2.86E+02±1.25E+02	1.40E+04±1.36E+04−	9.00E+02±4.25E+02
	cf_{18}	1.14E+02±1.97E+02−	2.94E+01±1.95E+01	1.66E+01±6.53E+00−	1.39E+01±9.31E+00	3.52E+02±4.95E+02−	5.94E+01±3.10E+01
	cf_{19}	4.48E+00±7.56E-01≈	4.48E+00±7.83E-01	4.36E+00±5.94E-01−	4.16E+00±6.71E-01	6.31E+00±1.15E+01−	4.58E+00±7.50E-01
	cf_{20}	3.11E+03±3.01E+03−	1.22E+01±4.65E+00	1.16E+01±3.51E+00−	8.51E+00±1.90E+00	1.39E+02±2.02E+02−	2.12E+01±1.08E+01
	cf_{21}	1.33E+04±4.12E+04−	1.40E+02±1.00E+02	2.74E+02±1.71E+02−	1.01E+02±8.28E+01	4.46E+03±7.23E+03−	2.38E+02±1.24E+02
	cf_{22}	1.44E+02±7.74E+01−	1.12E+02±7.46E+01	1.08E+02±7.15E+01−	7.21E+01±5.14E+01	1.54E+02±5.78E+01−	1.45E+02±8.03E+01
Composition Functions	cf_{23}	3.15E+02±4.01E-13≈	3.15E+02±3.59E-13	3.15E+02±4.01E-13≈	3.15E+02±3.73E-13	3.15E+02±2.24E-13≈	3.15E+02±2.45E-13
	cf_{24}	2.25E+02±3.60E+00≈	2.24E+02±1.85E+00	2.25E+02±2.56E+00≈	2.23E+02±7.89E-01	2.26E+02±2.79E+00≈	2.24E+02±9.64E-01
	cf_{25}	2.03E+02±1.13E+00≈	2.03E+02±4.16E-01	2.03E+02±5.31E-01≈	2.02E+02±3.67E-01	2.08E+02±2.54E+00≈	2.03E+02±1.62E+00
	cf_{26}	1.02E+02±1.39E+01≈	1.00E+02±4.27E-02	1.00E+02±4.02E-02≈	1.00E+02±3.64E-02	1.11E+02±3.24E+01−	1.00E+02±4.15E-02
	cf_{27}	3.35E+02±4.68E+01≈	3.34E+02±4.61E+01	3.62E+02±4.69E+01≈	3.70E+02±4.53E+01	4.20E+02±4.42E+01−	3.66E+02±4.28E+01
	cf_{28}	7.96E+02±4.63E+01≈	7.96E+02±4.57E+01	7.99E+02±2.68E+01≈	8.01E+02±3.54E+01	8.93E+02±3.46E+01−	8.18E+02±4.69E+01
	cf_{29}	8.28E+02±3.27E+02−	6.12E+02±2.00E+02	8.13E+02±7.12E+01−	5.43E+02±2.29E+02	1.10E+03±2.16E+02−	6.12E+02±1.67E+02
	cf_{30}	1.66E+03±7.61E+02−	1.02E+03±4.21E+02	1.40E+03±5.06E+02−	7.83E+02±3.71E+02	1.48E+03±5.40E+02−	9.75E+02±4.91E+02
+		0		1		3	
−		17		13		21	
≈		13		16		6	

TABLE S-IV
EXPERIMENTAL RESULTS OF JADE, ACOS-JADE, JDE, ACOS-JDE, SADE, AND ACOS-SADE OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 50D FROM IEEE CEC2014 USING 500,000 FES.

Test Functions with 50D from IEEE CEC2014		JADE Mean Error±Std Dev	ACoS-JADE Mean Error±Std Dev	jDE Mean Error±Std Dev	ACoS-jDE Mean Error Std Dev	SaDE Mean Error±Std Dev	ACoS-SaDE Mean Error±Std Dev
Unimodal Functions	cf_1	**1.45E+04±9.43E+03**−	**0.00E+00±0.00E+00**	4.58E+05±2.03E+05−	2.94E+02±1.30E+03	9.38E+05±2.96E+05−	4.63E+03±6.03E+03
	cf_2	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	**9.19E-09±1.83E-08**−	**0.00E+00±0.00E+00**	3.65E+03±4.02E+03−	2.90E+02±6.17E+02
	cf_3	**3.96E+03±2.41E+03**−	**0.00E+00±0.00E+00**	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	**3.04E+03±1.64E+03**−	**0.00E+00±0.00E+00**
Simple Multimodal Functions	cf_4	2.35E+01±4.12E+01−	1.10E+01±3.07E+01	8.70E+01±1.92E+01−	4.45E+01±3.94E+01	9.34E+01±3.89E+01−	2.84E+01±3.24E+01
	cf_5	2.03E+01±2.81E-02≈	2.03E+01±3.68E-02	2.04E+01±3.30E-02≈	2.04E+01±3.39E-02	2.07E+01±4.62E-02≈	2.05E+01 ±3.40E-02
	cf_6	1.56E+01±6.56E+00−	1.17E+01±7.19E+00	8.88E+00±7.14E+00−	6.34E+00±5.31E+00	1.77E+01±3.55E+00−	7.77E+00±2.56E+00
	cf_7	2.84E-03±6.74E-03−	8.69E-04±2.73E-03	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	1.43E-02±1.36E-02−	3.28E-03±4.56E-03
	cf_8	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00	**1.46E+00±1.78E+00**−	**0.00E+00±0.00E+00**
	cf_9	5.15E+01±7.85E+00≈	5.28E+01±6.61E+00	9.15E+01±9.51E+00−	8.92E+01±1.12E+01	8.78E+01±1.45E+01+	1.01E+02±2.16E+01
	cf_{10}	1.17E-02±1.33E-02−	9.79E-03±1.09E-02	7.34E-04±2.96E-03+	2.69E-03±5.18E-03	1.57E+00±1.11E+00−	1.06E-01±1.35E-01
	cf_{11}	3.84E+03±3.04E+02≈	3.93E+03±3.76E+02	5.22E+03±3.62E+02≈	5.26E+03±3.88E+02	6.49E+03±1.70E+03−	6.07E+03±4.09E+02
	cf_{12}	2.50E-01±3.42E-02≈	2.64E-01±3.71E-02	4.93E-01±5.40E-02≈	5.02E-01±5.75E-02	1.10E+00±1.10E-01−	6.27E-01±8.33E-02
	cf_{13}	3.29E-01±4.25E-02−	3.16E-01±4.45E-02	3.84E-01±4.45E-02−	3.68E-01±3.80E-02	4.26E-01±5.82E-02−	4.01E-01±6.71E-02
	cf_{14}	3.04E-01±8.56E-02≈	2.98E-01±9.03E-02	3.26E-01±5.50E-02−	2.95E-01±3.04E-02	3.09E-01±3.72E-02−	2.93E-01±2.95E-02
	cf_{15}	7.17E+00±8.18E-01+	7.35E+00±8.60E-01	1.20E+01±1.28E+00≈	1.19E+01±1.39E+00	1.46E+01±4.25E+00+	1.74E+01±2.32E+00
	cf_{16}	1.77E+01±3.71E-01≈	1.78E+01±5.19E-01	1.82E+01±3.72E-01≈	1.84E+01±4.20E-01	2.01E+01±3.19E-01−	1.88E+01±3.95E-01
Hybrid Functions	cf_{17}	2.40E+03±6.31E+02≈	2.42E+03±5.03E+02	2.16E+04±1.32E+04−	2.09E+03±5.36E+02	5.72E+04±3.41E+04−	2.12E+03±5.72E+02
	cf_{18}	1.74E+02±5.03E+01≈	1.88E+02±4.92E+01	5.91E+02±7.38E+02−	1.22E+02±4.01E+01	5.82E+02±5.63E+02−	4.93E+02±4.00E+01
	cf_{19}	1.30E+01±6.01E+00−	1.07E+01±3.80E+00	1.30E+01±4.48E+00≈	1.20E+01±2.89E+00	1.39E+01±6.45E+00+	1.67E+01±1.04E+01
	cf_{20}	8.27E+03±6.67E+03−	2.38E+02±7.09E+01	4.86E+01±1.64E+01−	4.43E+01±1.89E+01	8.79E+02±6.50E+02−	1.54E+02±7.20E+01
	cf_{21}	1.25E+03±3.07E+02+	1.41E+03±4.21E+02	8.53E+03±7.94E+03−	9.80E+02±2.91E+02	6.25E+04±3.33E+04−	1.18E+03±5.07E+02
	cf_{22}	4.81E+02±1.58E+02−	4.32E+02±1.54E+02	5.40E+02±1.12E+02−	4.30E+02±1.50E+02	4.80E+02±1.43E+02+	5.04E+02±1.08E+02
Composition Functions	cf_{23}	3.44E+02±4.26E-13≈	3.44E+02±4.98E-13	3.44E+02±4.17E-13≈	3.44E+02±2.87E-13	3.44E+02±2.85E-13≈	3.44E+02±2.87E-13
	cf_{24}	2.74E+02±1.66E+00≈	2.74E+02±2.20E+00	2.68E+02±2.17E+00≈	2.67E+02±2.45E+00	2.75E+02±3.44E+00≈	2.69E+02±4.68E+00
	cf_{25}	2.16E+02±7.03E+00−	2.09E+02±2.64E+00	2.07E+02±1.47E+00≈	2.07E+02±1.44E+00	2.17E+02±8.71E+00−	2.10E+02±9.31E+00
	cf_{26}	1.00E+02±1.33E-01≈	1.00E+02±6.88E-02	1.00E+02±3.69E-02≈	1.00E+02±5.47E-02	1.94E+02±2.36E+01−	1.23E+02±4.26E+01
	cf_{27}	4.48E+02±5.03E+01+	4.77E+02±6.60E+01	4.48E+02±7.13E+01−	4.04E+02±6.10E+01	7.67E+02±6.81E+01−	5.69E+02±6.85E+01
	cf_{28}	1.18E+03±5.71E+01≈	1.17E+03±9.34E+01	1.09E+03±3.35E+01≈	1.13E+03±5.82E+01	1.41E+03±1.25E+02−	1.20E+03±6.72E+01
	cf_{29}	9.00E+02±6.73E+01−	8.72E+02±8.50E+01	1.04E+03±1.95E+02−	8.22E+02±5.41E+01	1.43E+03±3.82E+02−	9.32E+02±1.06E+02
	cf_{30}	9.71E+03±7.21E+02≈	9.89E+03±1.08E+03	8.70E+03±4.77E+02≈	8.88E+03±5.72E+02	1.19E+04±1.78E+03−	9.99E+03±9.46E+02
+		3		1		4	
−		12		14		23	
≈		15		15		3	

TABLE S-V
EXPERIMENTAL RESULTS OF CMA-ES, COJADE, JADE/EIG, CPI-JADE, AND ACOS-JADE OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 30D FROM IEEE CEC2014 USING 300,000 FES.

Test Functions with 30D from IEEE CEC2014		CMA-ES Mean Error±Std Dev	CoJADE Mean Error±Std Dev	JADE/eig Mean Error±Std Dev	CPI-JADE Mean Error±Std Dev	ACoS-JADE Mean Error±Std Dev
Unimodal Functions	cf_1	0.00E+00±0.00E+00≈	3.96E+01±1.27E+02−	1.00E+02±3.12E+02−	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_2	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_3	0.00E+00±0.00E+00≈	5.90E-01±6.44E-01−	2.30E-02±3.83E-02−	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
Simple Multimodal Functions	cf_4	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_5	2.00E+01±7.43E-05+	2.03E+01±1.09E-01≈	2.03E+01±4.46E-02≈	2.03E+01±3.68E-02≈	2.03E+01±6.08E-02
	cf_6	3.98E+01±1.04E+01−	7.67E+00±3.51E+00−	6.98E+00±4.06E+00−	3.44E+00±3.57E+00+	6.11E+00±3.54E+00
	cf_7	1.15E-03±3.95E-03−	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_8	3.94E+02±8.31E+01−	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_9	6.53E+02±1.70E+02−	2.66E+01±4.25E+00−	2.46E+01±4.80E+00≈	2.24E+01±5.33E+00≈	2.47E+01±3.89E+00
	cf_{10}	4.89E+03±7.73E+02−	7.57E-02±3.08E-02−	5.47E-01±1.55E-01−	3.83E-01±7.47E-02−	4.89E-03±9.84E-03
	cf_{11}	5.29E+03±7.24E+02−	1.76E+03±2.41E+02≈	1.83E+03±2.25E+02−	1.77E+03±2.55E+02≈	1.69E+03±1.88E+02
	cf_{12}	2.33E-01±2.25E-01+	2.93E-01±4.28E-02−	3.14E-01±5.91E-02−	3.95E-01±8.64E-02−	2.80E-01±3.76E-02
	cf_{13}	2.60E-01±7.65E-02−	2.19E-01±3.64E-02≈	2.23E-01±4.05E-02≈	2.04E-01±3.38E-02+	2.21E-01±3.64E-02
	cf_{14}	3.54E-01±7.26E-02−	2.31E-01±3.02E-02≈	2.31E-01±3.00E-02−	2.32E-01±3.35E-02≈	2.24E-01±3.04E-02
	cf_{15}	3.45E+00±7.93E-01−	3.23E+00±3.59E-01≈	3.27E+00±4.45E-01≈	3.26E+00±3.78E-01≈	3.16E+00±3.76E-01
	cf_{16}	1.42E+01±4.26E-01−	9.62E+00±2.93E-01−	9.76E+00±3.48E-01−	9.70E+00±2.79E-01−	9.41E+00±4.33E-01
Hybrid Functions	cf_{17}	1.63E+03±4.26E+02−	1.26E+03±3.76E+02−	1.42E+03±4.38E+02−	1.16E+03±3.81E+02−	4.11E+02±1.56E+02
	cf_{18}	1.42E+02±4.79E+01−	1.01E+02±3.33E+01−	9.73E+01±3.70E+01−	9.47E+01±3.42E+01−	2.94E+01±1.95E+01
	cf_{19}	1.01E+01±1.75E+00−	4.66E+00±8.20E-01−	4.59E+00±6.63E-01≈	4.89E+00±7.64E-01−	4.48E+00±7.83E-01
	cf_{20}	2.80E+02±9.48E+01−	5.55E+02±7.08E+02−	2.52E+02±2.57E+02−	1.12E+01±5.24E+00≈	1.22E+01±4.65E+00
	cf_{21}	9.91E+02±3.05E+02−	1.45E+03±5.33E+03−	6.95E+02±1.11E+03−	3.33E+02±1.54E+02−	1.40E+02±1.00E+02
	cf_{22}	2.87E+02±1.56E+02−	1.07E+02±6.93E+01≈	1.03E+02±6.94E+02≈	9.99E+01±6.09E+01≈	1.12E+02±7.46E+01
Composition Functions	cf_{23}	3.15E+02±4.20E-12≈	3.15E+02±4.01E-13≈	3.15E+02±4.01E-13≈	3.15E+02±4.01E-13≈	3.15E+02±3.59E-13
	cf_{24}	2.48E+02±6.72E+01−	2.24E+02±1.72E+00≈	2.25E+02±3.53E+00≈	2.24E+02±2.93E+00≈	2.24E+02±1.85E+00
	cf_{25}	2.03E+02±1.64E+00≈	2.03E+02±9.00E-01≈	2.03E+02±6.41E-01≈	2.03E+02±5.77E-01≈	2.03E+02±4.16E-01
	cf_{26}	1.35E+02±1.83E+02−	1.00E+02±4.71E-02≈	1.00E+02±4.53E-02≈	1.00E+02±2.92E-02≈	1.00E+02±4.27E-02
	cf_{27}	3.81E+02±5.84E+01−	3.41E+02±4.96E+01≈	3.45E+02±4.85E+01−	3.53E+02±5.03E+01−	3.34E+02±4.61E+01
	cf_{28}	3.69E+03±3.63E+03−	8.02E+02±3.81E+01≈	8.06E+02±3.69E+01−	8.02E+02±4.34E+01≈	7.96E+02±4.57E+01
	cf_{29}	7.91E+02±9.09E+01−	7.31E+02±1.26E+01−	7.30E+02±3.79E+01−	8.13E+02±7.12E+01−	6.12E+02±2.00E+02
	cf_{30}	2.25E+03±7.06E+02−	1.77E+03±8.13E+02−	1.68E+03±7.22E+02−	1.40E+03±7.24E+02−	1.02E+03±4.21E+02
+		2	0	0	2	
−		22	14	16	10	
≈		6	16	14	18	

TABLE S-VI
EXPERIMENTAL RESULTS OF NONAR-ACOS-JADE, JADE, HALF-ACOS-JADE, EIG-ACOS-JADE, AND ACOS-JADE OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 30D FROM IEEE CEC2014 USING 300,000 FES.

Test Functions with 30D from IEEE CEC2014		nonAr-ACoS-JADE Mean Error±Std Dev	JADE Mean Error±Std Dev	half-ACoS-JADE Mean Error±Std Dev	Eig-ACoS-JADE Mean Error±Std Dev	ACoS-JADE Mean Error±Std Dev
Unimodal Functions	cf_1	0.00E+00±0.00E+00≈	6.09E+02±1.18E+03−	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_2	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_3	0.00E+00±0.00E+00≈	9.86E-04±5.95E-03−	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
Simple Multimodal Functions	cf_4	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00
	cf_5	2.03E+01±4.03E-02≈	2.03E+01±3.23E-02≈	2.03E+01±6.08E-02≈	2.07E+01±3.02E-01−	2.03E+01±6.08E-02
	cf_6	6.60E+00±3.66E+00−	9.15E+00±2.21E+00−	6.70E+00±3.44E+00−	8.19E+00±4.90E+00−	6.11E+00±3.54E+00
	cf_7	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	9.17E-04±2.95E-03−	0.00E+00±0.00E+00
	cf_8	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	0.00E+00±0.00E+00≈	2.69E+01±5.05E+00−	0.00E+00±0.00E+00
	cf_9	2.41E+01±3.40E+00≈	2.62E+01±4.96E+00−	2.46E+01±4.42E+00≈	2.79E+01±5.73E+00−	2.47E+01±3.89E+00
	cf_{10}	6.93E-03±9.91E-03−	8.16E-03±1.18E-02−	5.64E-01±1.46E-01−	2.35E+03±2.35E+02−	4.89E-03±9.84E-03
	cf_{11}	1.72E+03±2.41E+02≈	1.67E+03±2.13E+02≈	1.82E+03±2.19E+02−	2.75E+03±2.75E+02−	1.69E+03±1.88E+02
	cf_{12}	2.85E-01±3.95E-02≈	2.67E-01±3.57E-02≈	2.95E-01±4.39E-02−	6.32E-01±1.42E-01−	2.80E-01±3.76E-02
	cf_{13}	2.19E-01±3.15E-02≈	2.20E-01±3.25E-02≈	2.18E-01±3.79E-02≈	2.32E-01±4.46E-02−	2.21E-01±3.64E-02
	cf_{14}	2.27E-01±3.29E-02≈	2.41E-01±3.18E-02−	2.34E-01±3.28E-02−	2.15E-01±6.23E-02≈	2.24E-01±3.04E-02
	cf_{15}	3.11E+00±4.27E-01≈	3.20E+00±4.55E-01≈	3.22E+00±3.49E-01≈	4.00E+00±6.87E-01−	3.16E+00±3.76E-01
	cf_{16}	9.52E+00±3.68E-01−	9.30E+00±4.61E-01≈	9.68E+00±3.43E-01−	1.08E+01±5.39E-01−	9.41E+00±4.33E-01
Hybrid Functions	cf_{17}	4.11E+02±1.56E+02≈	1.91E+04±1.08E+05−	3.87E+02±1.94E+02+	5.52E+02±2.33E+02−	4.11E+02±1.56E+02
	cf_{18}	9.21E+01±2.98E+01−	1.14E+02±1.97E+02−	2.04E+01±1.42E+01+	8.60E+01±2.63E+01−	2.94E+01±1.95E+01
	cf_{19}	4.79E+00±8.02E-01−	4.48E+00±7.56E-01≈	4.48E+00±7.72E-01≈	5.89E+00±1.11E+00−	4.48E+00±7.83E-01
	cf_{20}	2.29E+01±1.11E+01−	3.11E+03±3.01E+03−	1.18E+01±9.27E+00≈	4.69E+01±2.02E+01−	1.22E+01±4.65E+00
	cf_{21}	3.66E+02±1.99E+02−	1.33E+04±4.12E+04−	1.60E+02±8.74E+01−	4.51E+02±1.34E+02−	1.40E+02±1.00E+02
	cf_{22}	1.30E+02±6.55E+01−	1.44E+02±7.74E+01−	9.76E+01±6.22E+01+	1.48E+02±7.06E+01−	1.12E+02±7.46E+01
Composition Functions	cf_{23}	3.15E+02±4.01E-13≈	3.15E+02±4.01E-13≈	3.15E+02±4.01E-13≈	3.15E+02±4.01E-13≈	3.15E+02±3.59E-13
	cf_{24}	2.24E+02±1.81E+00≈	2.25E+02±3.60E+00≈	2.24E+02±2.59E+00≈	2.27E+02±4.62E+00≈	2.24E+02±1.85E+00
	cf_{25}	2.03E+02±5.71E-01≈	2.03E+02±1.13E+00≈	2.02E+02±2.99E-01≈	2.03E+02±5.66E-01≈	2.03E+02±4.16E-01
	cf_{26}	1.06E+02±2.37E+01−	1.02E+02±1.39E+01≈	1.00E+02±3.50E-02≈	1.00E+02±5.39E-02≈	1.00E+02±4.27E-02
	cf_{27}	3.43E+02±4.75E+01≈	3.35E+02±4.68E+01≈	3.49E+02±4.80E+01−	3.82E+02±4.18E+01−	3.34E+02±4.61E+01
	cf_{28}	8.01E+02±4.06E+01≈	7.96E+02±4.63E+01≈	8.06E+02±3.56E+01−	8.58E+02±6.96E+01−	7.96E+02±4.57E+01
	cf_{29}	1.80E+05±1.28E+06−	8.28E+02±3.27E+02−	7.07E+02±8.30E+02−	4.60E+05±2.31E+06−	6.12E+02±2.00E+02
	cf_{30}	1.53E+03±7.69E+02−	1.66E+03±7.61E+02−	1.13E+03±4.87E+02−	1.27E+03±6.65E+02−	1.02E+03±4.21E+02
+		0	0	3	0	
−		11	13	11	21	
≈		19	17	16	9	

TABLE S-VII
EXPERIMENTAL RESULTS OF BA, ACOS-BA, TLBO, AND ACOS-TLBO OVER 51 INDEPENDENT RUNS ON 30 TEST FUNCTIONS WITH 30D FROM IEEE CEC2014 USING 300,000 FES.

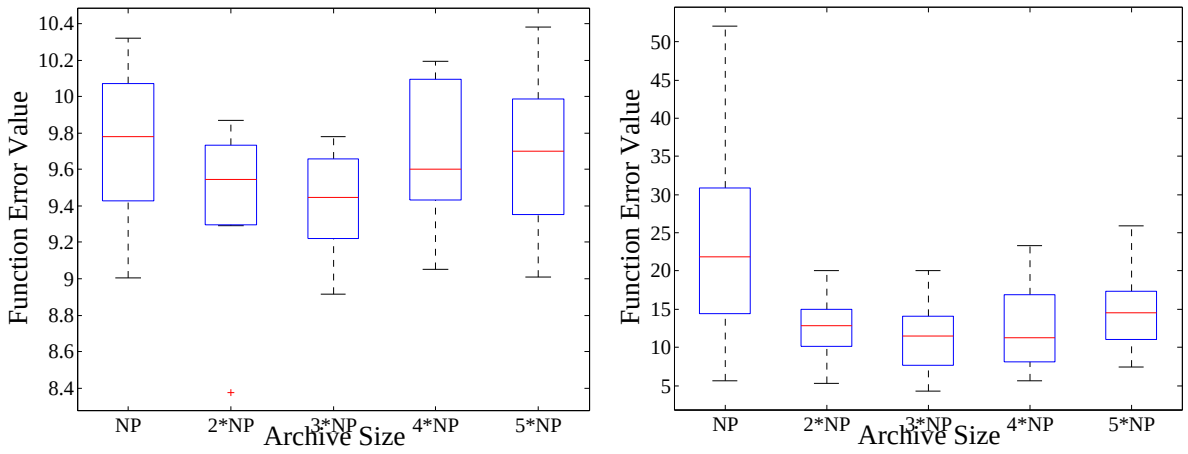
Test Functions with 30D from IEEE CEC2014		BA Mean Error±Std Dev	ACoS-BA Mean Error±Std Dev	TLBO Mean Error Std Dev	ACoS-TLBO Mean Error±Std Dev
Unimodal Functions	cf_1	1.42E+05±9.29E+04−	2.36E+04±2.04E+04	2.34E+07±6.90E+06−	1.60E+02±9.16E+01
	cf_2	1.19E+04±1.43E+04−	9.38E+03±9.92E+03	3.58E+07±7.23E+06−	3.65E+04±1.89E+04
	cf_3	5.00E-03±3.99E-03−	1.31E-05±2.09E-05	4.43E+04±8.48E+03−	1.89E-01±6.23E-02
Simple Multimodal Functions	cf_4	4.52E+01±4.18E+01−	6.32E+00±1.90E+01	1.62E+02±2.42E+01−	9.77E+01±2.58E+01
	cf_5	2.02E+01±1.94E-01≈	2.01E+01±1.82E-01	2.09E+01±5.86E-02≈	2.08E+01±5.17E-02
	cf_6	4.48E+01±2.32E+00−	4.37E+01±2.48E+00	3.29E+01±1.73E+00−	2.83E+01±1.85E+00
	cf_7	1.57E-02±1.31E-02−	1.11E-02±1.20E-02	1.08E+00±2.59E-02−	4.59E-01±8.13E-02
	cf_8	2.36E+02±4.89E+01≈	2.40E+02±5.01E+01	2.29E+02±1.04E+01−	1.83E+02±2.11E+01
	cf_9	3.16E+02±5.41E+01−	3.06E+02±5.10E+01	2.67E+02±1.49E+01−	2.04E+02±1.89E+01
	cf_{10}	4.53E+03±8.49E+02≈	4.58E+03±7.45E+02	5.91E+03±3.08E+02−	4.99E+03±3.46E+02
	cf_{11}	5.12E+03±8.74E+02−	4.57E+03±8.60E+02	6.90E+03±3.36E+02−	5.46E+03±4.16E+02
	cf_{12}	1.63E+00±6.18E-01−	1.47E+00±6.12E-01	2.43E+00±2.71E-01−	1.83E+00±2.09E-01
	cf_{13}	7.25E-01±1.41E-01−	7.04E-01±1.68E-01	5.69E-01±6.94E-02−	4.13E-01±5.56E-02
	cf_{14}	7.68E-01±3.63E-01−	4.73E-01±3.14E-01	4.27E-01±1.82E-01−	3.29E-01±1.92E-01
	cf_{15}	7.89E+02±5.23E+02−	5.79E+02±4.29E+02	1.00E+02±4.05E+01−	2.56E+01±2.25E+00
	cf_{16}	1.35E+01±4.62E-01≈	1.35E+01±4.73E-01	1.28E+01±1.77E-01≈	1.24E+01±2.53E-01
Hybrid Functions	cf_{17}	3.82E+03±1.86E+03−	1.89E+03±4.74E+02	8.02E+05±3.22E+05−	1.57E+03±2.20E+02
	cf_{18}	1.25E+04±9.72E+03−	3.68E+02±9.41E+01	1.83E+04±3.13E+04−	8.74E+01±1.43E+01
	cf_{19}	3.51E+01±2.80E+01−	2.87E+01±2.30E+01	1.26E+01±1.56E+00−	9.37E+00±9.98E-01
	cf_{20}	5.01E+02±1.80E+02−	3.79E+02±1.16E+02	6.92E+03±2.23E+03−	7.47E+01±1.46E+01
	cf_{21}	2.67E+03±1.14E+03−	1.52E+03±3.72E+02	2.27E+05±9.07E+04−	1.01E+03±1.60E+02
	cf_{22}	1.16E+03±3.37E+02≈	1.10E+03±3.94E+02	4.62E+02±9.13E+01−	2.52E+02±8.29E+01
Composition Functions	cf_{23}	3.15E+02±1.70E-08≈	3.15E+02±3.19E-09	3.15E+02±4.78E-02−	2.16E+02±4.37E+00
	cf_{24}	2.59E+02±3.36E+01≈	2.55E+02±2.34E+01	2.48E+02±3.12E+00−	2.41E+02±7.84E+00
	cf_{25}	2.13E+02±2.05E+01≈	2.09E+02±6.95E+00	2.09E+02±1.92E+00−	2.02E+02±3.95E+00
	cf_{26}	1.32E+02±7.88E+01≈	1.32E+02±7.01E+01	1.00E+02±7.12E-02≈	1.00E+02±7.20E-02
	cf_{27}	1.31E+03±4.34E+02≈	1.29E+03±4.31E+02	5.26E+02±4.44E+01−	4.17E+02±9.24E+01
	cf_{28}	1.93E+03±4.59E+02≈	1.95E+03±4.84E+02	9.97E+02±1.11E+02≈	1.03E+03±1.59E+02
	cf_{29}	1.54E+07±1.64E+07−	9.54E+06±1.36E+07	2.88E+06±5.52E+06≈	2.83E+06±5.45E+06
	cf_{30}	4.31E+03±2.76E+03−	3.37E+03±1.36E+03	3.53E+03±1.12E+03−	2.13E+03±6.40E+02
+		0		0	
−		19		25	
≈		11		5	

TABLE S-VIII
RUNTIME OF PSO-W, ACOS-PSO-W, PSO-CF, ACOS-PSO-CF, JADE, ACOS-JADE, JDE, ACOS-JDE, SADE, AND ACOS-SADE ON 30 TEST FUNCTIONS WITH 30D FROM IEEE CEC2014 USING 300,000 FES.

Test Functions with 30D from IEEE CEC2014		PSO-w (second)	ACoS-PSO-w (second)	PSO-cf (second)	ACoS-PSO-cf (second)	JADE (second)	ACoS-JADE (second)	jDE (second)	ACoS-jDE (second)	SaDE (second)	ACoS-SaDE (second)
Unimodal Functions	cf_1	2.2142	3.8724	2.3895	3.8101	3.3886	7.0318	3.0132	5.0854	7.9106	10.4659
	cf_2	1.8028	3.4501	1.7901	3.3946	2.7497	6.4724	2.6458	4.7716	7.7328	10.2672
	cf_3	1.8355	3.4861	1.7733	3.3074	2.9881	6.2536	2.5854	4.7484	7.6045	9.9706
Simple Multimodal Functions	cf_4	1.7991	3.4287	1.8231	3.4503	2.8028	6.3221	2.6797	4.5554	7.4201	9.7892
	cf_5	2.0296	3.7182	1.9651	3.6389	3.2044	4.8771	3.1222	5.0557	7.1311	9.8812
	cf_6	17.8331	20.9827	16.7533	19.7588	19.6648	21.4436	19.2908	20.4357	22.8878	27.3678
	cf_7	2.0408	3.6955	2.1028	3.4738	2.8274	6.4546	2.6883	4.8135	7.7694	10.1482
	cf_8	1.7647	3.4323	1.6579	3.4091	2.7498	5.9133	2.5043	4.6337	7.5304	9.9611
	cf_9	1.9354	3.6409	1.8324	3.5451	3.0866	5.0762	2.8706	5.0569	7.1942	9.6662
	cf_{10}	2.4234	4.1563	2.3496	4.1261	3.5147	6.0472	3.1431	5.3699	7.7905	10.3124
	cf_{11}	2.5831	4.3876	2.4547	4.2967	3.8424	5.8652	3.7066	5.7097	7.9897	10.2332
	cf_{12}	4.8451	6.9136	4.8229	6.7826	6.0392	8.3581	5.9295	8.4344	10.0151	12.7009
	cf_{13}	1.8233	3.4658	1.7786	3.3718	2.9269	4.4212	2.7014	4.5587	6.6291	9.1342
	cf_{14}	1.8005	3.4181	1.7821	3.3554	2.8561	4.4291	2.6962	4.5533	6.5993	8.6736
	cf_{15}	1.9537	3.6559	1.9242	3.5005	3.0394	4.9871	2.6782	4.6438	6.9707	9.3157
	cf_{16}	2.0467	3.7589	1.9877	3.6301	3.2234	5.0719	3.0945	4.9776	7.2787	9.4789
Hybrid Functions	cf_{17}	2.3104	3.9045	2.3388	3.8691	3.1569	5.3063	3.0816	4.8847	7.6574	9.9113
	cf_{18}	1.9401	3.5804	2.0079	3.5527	2.9495	4.9686	2.7377	4.5271	7.1801	9.6157
	cf_{19}	4.8291	7.4193	4.7044	7.3484	6.4186	8.8967	6.1631	8.6438	10.6072	13.2464
	cf_{20}	1.9966	3.6338	1.9657	3.5297	3.0002	5.0601	2.7634	4.5224	7.0747	9.5677
	cf_{21}	2.2228	3.8929	2.3242	3.7191	3.2053	5.4611	3.0374	4.8535	7.4862	9.9057
	cf_{22}	2.4539	4.2028	2.4351	4.1052	3.6891	5.7394	3.3799	5.4157	7.7426	10.4195
Composition Functions	cf_{23}	5.2665	7.4386	5.2572	7.4443	6.4335	10.2934	6.1291	8.9888	11.3682	14.2865
	cf_{24}	4.1893	6.2041	4.1255	6.0506	5.1305	7.3717	4.8827	7.2648	8.9745	11.6183
	cf_{25}	4.8157	6.7232	4.8157	6.6979	5.8016	9.8592	5.6377	8.1103	10.6325	13.2666
	cf_{26}	22.6681	25.6779	22.4181	25.6285	24.3567	26.4984	23.7301	26.3194	28.1501	31.6183
	cf_{27}	22.5864	24.9456	21.1078	24.5395	23.1577	24.9597	21.1501	25.3616	26.0935	31.9194
	cf_{28}	6.3173	8.7162	6.3336	8.7289	7.6282	10.3412	7.2752	9.9322	11.9131	14.6997
	cf_{29}	6.8505	10.4318	7.0055	10.5782	8.3378	11.4457	8.6174	10.9918	13.2429	16.3783
	cf_{30}	4.5956	6.6341	4.6865	6.5721	5.6371	8.6383	5.6018	7.8296	10.2773	12.6853
Average Runime		4.7924	6.7622	4.6904	6.6405	5.9269	8.4621	5.6512	7.8349	10.1618	12.8835
Average Extra Runtime		1.9698		1.9501		2.5352		2.1837		2.7217	

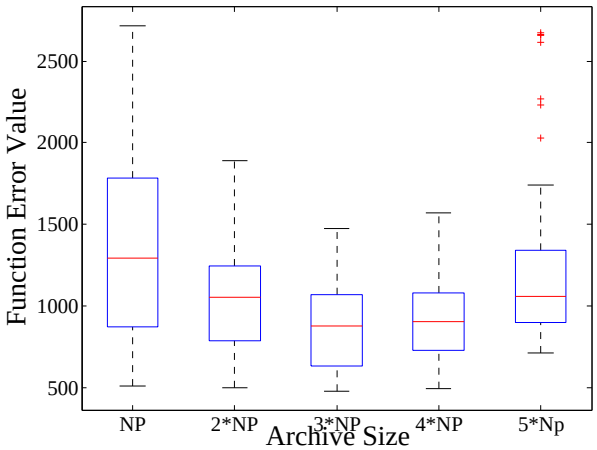
TABLE S-IX
RUNTIME OF PSO-W, ACOS-PSO-W, PSO-CF, ACOS-PSO-CF, JADE, ACOS-JADE, JDE, ACOS-JDE, SADE, AND ACOS-SADE ON 30 TEST FUNCTIONS WITH 50D FROM IEEE CEC2014 USING 500,000 FES.

Test Functions with 50D from IEEE CEC2014		PSO-w (second)	ACoS-PSO-w (second)	PSO-cf (second)	ACoS-PSO-cf (second)	JADE (second)	ACoS-JADE (second)	jDE (second)	ACoS-jDE (second)	SaDE (second)	ACoS-SaDE (second)
Unimodal Functions	cf_1	5.2851	9.5367	5.4502	9.4047	7.1422	14.7701	6.9303	12.8697	15.1198	20.5759
	cf_2	4.2332	8.4885	4.1111	8.1127	5.8121	13.6929	5.9791	11.9432	13.986	19.7086
	cf_3	4.1933	8.4791	3.9686	8.1145	6.4448	13.6418	5.8346	11.9587	13.758	19.4487
Simple Multimodal Functions	cf_4	4.1449	8.2917	4.1463	8.4678	6.5183	13.7654	6.1575	12.2024	14.2523	19.7847
	cf_5	4.7193	9.0402	4.5959	8.7729	7.0949	11.9777	7.1862	12.9257	13.5849	19.8751
	cf_6	49.6886	56.6993	48.1424	54.4512	51.1161	57.7224	50.9719	54.0292	58.9581	65.0065
	cf_7	4.8041	9.1333	4.3877	8.4499	5.9057	14.1342	5.8287	12.4163	14.6453	20.0929
	cf_8	3.7759	7.9644	3.4189	7.8126	5.3699	12.4307	5.0886	11.5266	13.775	19.2783
	cf_9	4.6789	8.7991	4.2335	8.5548	6.6893	11.6927	6.2278	12.3079	13.7503	19.5798
	cf_{10}	5.5222	9.8839	5.2883	9.7324	7.6872	13.5845	6.9255	13.2394	14.6066	20.4837
	cf_{11}	6.5415	10.9008	6.0839	10.6675	8.7899	14.0852	8.9508	14.7223	15.7978	21.9198
	cf_{12}	12.8893	17.7061	12.4262	17.4444	15.1317	20.5847	14.8504	22.0975	22.1256	28.7203
	cf_{13}	4.0718	8.3992	4.0044	8.1753	6.0442	10.1349	5.9759	11.9562	12.8343	18.3817
	cf_{14}	4.1043	8.3078	4.0331	8.0798	5.9245	10.2801	5.8887	11.7078	12.4708	18.2361
	cf_{15}	4.6523	8.8733	4.4801	8.5269	6.4959	11.5836	5.9439	11.9709	13.1504	19.1214
	cf_{16}	4.8173	9.1135	4.6508	8.9176	7.0584	11.8899	7.2277	13.0044	13.8892	20.1175
Hybrid Functions	cf_{17}	5.4829	9.6939	5.6649	9.5823	7.1815	14.6118	7.1746	12.7941	15.2678	20.6719
	cf_{18}	4.4876	8.6565	4.4833	8.5788	6.2662	12.3668	5.8823	11.7213	13.4671	19.3159
	cf_{19}	13.2878	18.6763	12.4123	18.3133	15.5912	22.7477	15.2789	22.5083	23.6379	29.1993
	cf_{20}	4.6144	8.8146	4.4165	8.6001	6.3337	11.6084	6.0314	11.7596	13.8077	19.2949
	cf_{21}	5.2315	9.3426	5.3638	9.2568	7.5373	13.3399	6.8743	12.5068	14.7619	20.1605
	cf_{22}	5.8024	10.2609	5.7874	10.1903	8.3072	13.4967	7.8257	14.0026	15.2748	21.3178
Composition Functions	cf_{23}	14.6306	19.6574	14.7443	19.5975	16.8101	25.8235	16.4084	24.6849	25.3783	32.2466
	cf_{24}	10.9654	15.9341	10.5038	15.5605	12.4776	21.9365	12.2149	19.8828	21.1163	27.3674
	cf_{25}	13.0102	17.9162	13.0076	17.9137	14.6446	23.8497	14.7218	22.1201	23.1034	29.3759
	cf_{26}	63.8244	70.8617	63.3021	69.9452	65.9136	71.7998	65.5865	73.7435	73.9766	81.8428
	cf_{27}	62.8366	69.5244	61.3084	68.6068	61.1054	67.0864	61.3795	67.8372	68.6311	78.1318
	cf_{28}	17.9415	23.6857	17.8963	23.6539	20.9666	27.0057	20.3048	27.9469	28.0773	35.3722
	cf_{29}	19.2817	26.9251	19.1166	26.8671	22.4686	30.1808	22.5619	29.9646	30.9617	38.5667
	cf_{30}	12.3805	17.2641	12.5688	17.2178	14.8115	21.7373	14.4585	21.5516	22.4387	28.8285
Average Runime		12.7299	17.5611	12.4665	17.2523	14.6547	21.1187	14.4224	20.7968	22.2202	28.4007
Average Extra Runtime		4.8312		4.7858		6.4640		6.3744		6.1805	



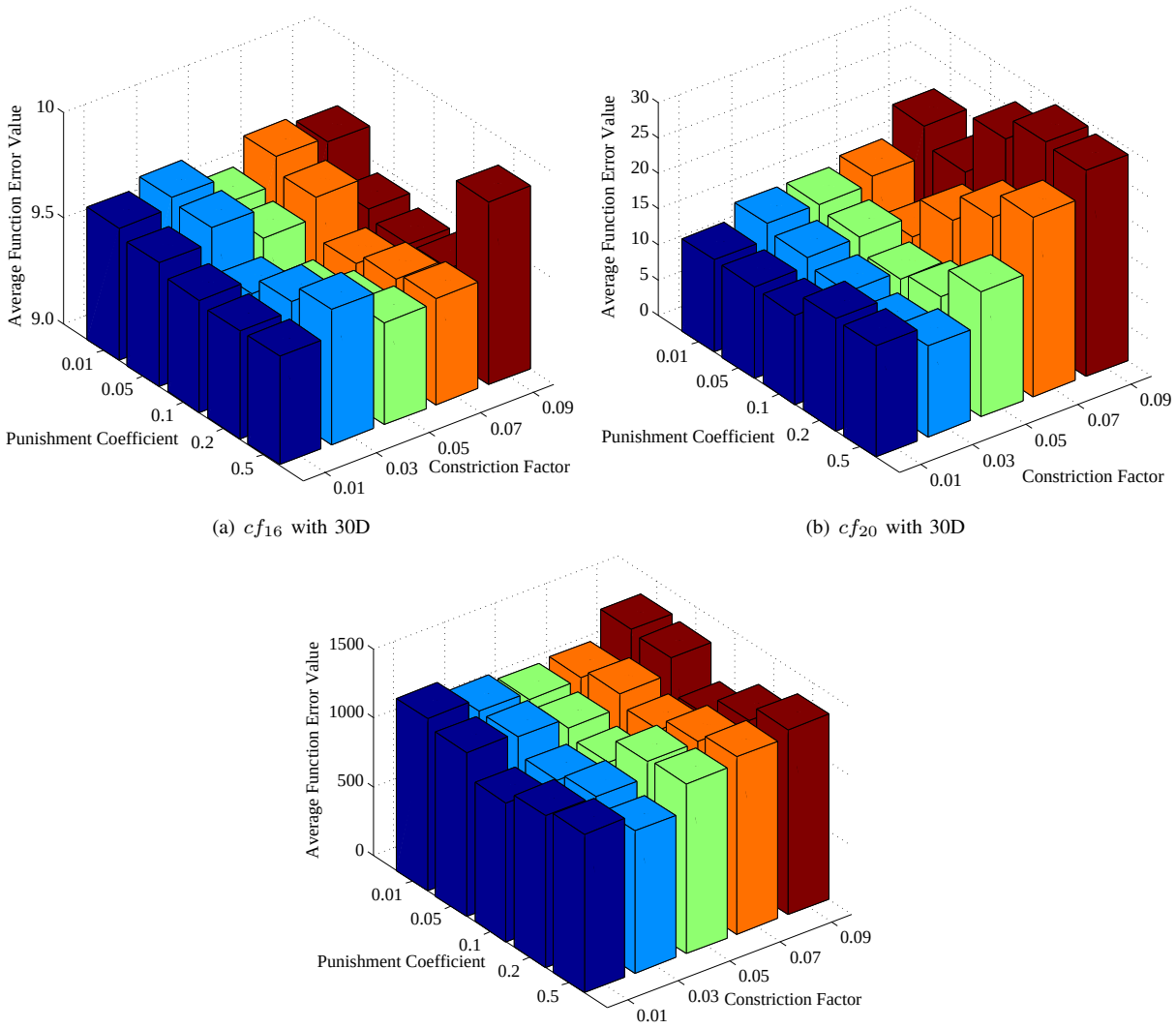
(a) cf_{16} with 30D

(b) cf_{20} with 30D



(c) cf_{30} with 30D

Fig. S-1. Box plots of the function error values derived from ACoS-JADE with different archive size on cf_{16} with 30D, cf_{20} with 30D, and cf_{30} with 30D.



(a) cf_{16} with 30D

(b) cf_{20} with 30D

(c) cf_{30} with 30D

Fig. S-2. Average function error values provided by ACoS-JADE with different combinations of constriction factor ε and punishment efficient η on cf_{16} with 30D, cf_{20} with 30D, and cf_{30} with 30D.