Information Sciences 394–395 (2017) 273–298

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Vector coevolving particle swarm optimization algorithm

Qingke Zhang[a], Weiguo Liu[a,*], Xiangxu Meng[a,*], Bo Yang[b,*], Athanasios V. Vasilakos[c]

[a] School of Computer Science and technology, Engineering Research Center of Digital Media Technology, Ministry of Education, Shandong University, Jinan, 250101, China
[b] Shandong Provincial Key Laboratory of Network based Intelligent Computing, University of Jinan, Jinan, 250022, China
[c] Department of Computer Science, Electrical and Space Engineering Luleå University of Technology, SE-931 87 Skellefteå, Sweden

ARTICLE INFO

Article history:
Received 23 December 2015
Revised 25 January 2017
Accepted 30 January 2017
Available online 3 February 2017

Keywords:
Particle swarm optimization
Coevolving evolution
Vector partition
Centralized learning
Decentralized learning

ABSTRACT

In this paper, we propose a novel vector coevolving particle swarm optimization algorithm (VCPSO). In VCPSO, the full dimension of each particle is first randomly partitioned into several sub-dimensions. Then, we randomly assign either one of our newly designed scalar operators or learning operators to update the values in each sub-dimension. The scalar operators are designed to enhance the population diversity and avoid premature convergence. In addition, the learning operators are designed to enhance the global and local search ability. The proposed algorithm is compared with several other classical swarm optimizers on thirty-three benchmark functions. Comprehensive experimental results show that VCPSO displays a better or comparable performance compared to the other algorithms in terms of solution accuracy and statistical results.

1. Introduction

Optimization is an important area in scientific research. As many real-world optimization problems such as nonlinear optimal control, text clustering, DNA sequence compression, distribution network design are becoming increasingly more complex, there is high demand for more efficient optimization algorithms. Optimization problems can generally be classified into two categories: unconstrained problems and constrained problems. The constrained problems can be formulated as follows:

$$min\ \sigma = f(X), X \in S, \quad S = \{X | g_i(X) \leq 0, i = 1, \ldots, m\}$$

where $\sigma = f(X)$ is the objective function, and $g_i(X)$ is the constraints function. m is the number of constraint functions, and X is a D-dimensional vector. The constrained problems can be formulated as D-dimensional minimization problems in Euclidean n-space without any constrained function as follows:

$$min\ \sigma = f(X), X \in S, X = \{x_1, x_2, \ldots, x_D\}, S \subset R^n$$

where $\sigma = f(X)$ is the objective function, and X is a D-dimensional vector. Algorithms for solving these two optimization categories can be divided into deterministic and random optimization algorithms. Traditional deterministic algorithms, such as gradient-based algorithms use specific rules to move from one solution to the other. These methods have been proven

* Corresponding authors.
E-mail addresses: tsingke@hotmail.com (Q. Zhang), weiguo.liu@sdu.edu.cn, liuweiguoemail@gmail.com (W. Liu), mxx@sdu.edu.cn (X. Meng), yangbo@ujn.edu.cn (B. Yang), th.vasilakos@gmail.com (A.V. Vasilakos).

http://dx.doi.org/10.1016/j.ins.2017.01.038
0020-0255/

to be inefficient and have poor solution quality when solving optimization problems with nonlinear, high dimensional, and discontinuous features. The stochastic optimization search methods are used to tackle complex problems that are non differentiable, multi-modal and have multiple objectives while lacking smoothness. Nature inspired meta-heuristics are currently the most powerful tools for optimization among stochastic algorithms. Examples of such heuristics methods include genetic algorithms (GA), genetic programming (GP), simulated annealing (SA), particle swarm optimization (PSO), ant-colony optimization (ACO), differential evolution (DE), evolution strategies (ES), evolutionary programming (EP) estimation of distribution algorithm (EDA), and other hybrid methods. Among them, PSO is a computationally effective algorithm that was originally developed by Kennedy and Eberhart in 1995. It has been widely used in practice and turned out to be a competitor in the field of numerical optimization because of its high efficiency and low memory usage.

PSO is a global stochastic method that was designed to simulate the social behavior of bird flocks or biological groups [18]. Each particle of the swarm represents a set of optimization parameters. The set of parameters can be viewed as a potential solution in a multidimensional search space. The aim of the algorithm is to converge to an optimum value for each parameter. Although PSO has many advantages over other algorithms, it tends to converge at local optima in the early stage when it is used to solve complex problems . In addition, when the searching space dimension is high, the convergence speed of PSO and most of its variants showed slow convergence [12]. Therefore, accelerating the convergence speed and avoiding the local optima have become the two most important and appealing goals in particle swarm optimization research. Moreover, when solving high-dimensional or large-scale optimization problems, a fully dimensional learning strategy may cause poor solutions because the value of one dimension or the values of some parts of the full dimensions [22]. This is caused by both the existence of local optimal solutions and the degeneracy of particle velocities, but another obvious reason for these phenomena can be attributed to the independent full dimensional learning approach and the limited number of learning individuals in position updating equations. Based on the analysis, we proposed a novel vector coevolving particle swarm optimization algorithm (VCPSO). We partitioned the full dimension of a particle into several segments and then optimize each segment by different operators (or strategies). The operators co-evolve with each other and optimize each segment independently. The proposed VCPSO is remarkably different from the classical swarm optimization algorithms. The main contributions of our work can be summarized in the following:

- A vector partition technique are designed to partition the full dimension of a particle into several segments randomly and then each of segments is optimized independently by one of the randomly selected operators. The two-level randomized mechanism can help strengthen particles intelligence and prevent them from trapping in local optima.
- Four scalar operators are introduced. The scalar operators contain four operators, namely the increasing operator, decreasing operator, hill operator and lake operator. They are designed based on the inner sub-dimensions. These operators can help to enrich the population diversity.
- Two learning operators are designed. In order to enhance the global and local search ability, we have designed two novel learning operators, namely, centralized operator, decentralized operator. The centralized operator is designed to enhance the global search ability by learning the mean position of several top ranked particles. While the decentralized operator is designed to enhance the local search ability by learning from the better individual of two random selected particles.

These operators coevolved the sub-dimensions with each other and can be classified into two categories: learning operators and scalar operators. We acquired these operators motivated by human social learning models, such as the advantage study model by learning from several best individuals of a group (centralized operator) ; the specialty study model by learning from one individual of a group (decentralized operator) ;the self-learning study model by rectifying some shortcomings of its own self (scalar operators).

The rest of this paper is organized as follows. Section 2 introduces the original PSO and reviews its typical variants. Section 3 provides a detailed description of the vectorized coevolving particle swarm optimization algorithm. In Section 4, experiments are conducted as follows: first, the test benchmark functions are introduced. second, we studied the effect of the combination of different operators and the minimum partition interval for each segment on different dimensions. third, we compared the performance of with PSO variants and some other evolution algorithms, such as differential evolution algorithms in terms of accuracy and statistical results. Furthermore, the computational complexity of these compared algorithms and discusses are provided. Conclusions are given in the last section.

2. PSO and PSO variants

In this section, we make a brief review of PSO and some of its variants. These PSO variants were classified into four categories: parameter modification-based algorithms, neighborhood topology-based algorithms, learning strategies-based algorithms, and hybridized method-based algorithms.

2.1. PSO algorithm

PSO is a global optimization method applied to find the optimal solution X^{opt} of objective function f. The desired optimum is generally a minimum value problem. Each swarm has a population of N particles, and each particle is expressed by two vectors: position vector X_i and velocity vector V_i. The position vector X_i represents a potential solution to the optimized

problem, and the velocity vector V_i reflects the position-changing speed. When searching a D-dimensional problem, the particle is represented as $X_i = (x_i^1, \ldots, x_i^D)$, and the velocity is represented as $V_i = (v_i^1, \cdots, v_i^D)$. During the space searching procedure, each particle maintains a memory of its previous historical best position $Pbest_i = (p_i^1, \cdots, p_i^D)$. These particles search along the entire space of dimension by moving with a certain velocity to find the global best position $Gbest_i = (g_i^1, \cdots, g_i^D)$. To search for the optimal solution, the particle updates its position and velocity iteratively according to the Eq. (1) :

$$\begin{cases} V_i^D(t+1) = V_i^D(t) + c_1 r_1 (Pbest_i^D(t) - X_i^D(t)) + c_2 r_2 (Gbest_i^D(t) - X_i^D(t)) \\ X_i^D(t+1) = X_i^D(t) + V_i^D(t+1) \end{cases} \tag{1}$$

where t is the time step, r_1 and r_2 are random numbers uniformly distributed in the range of 0 and 1. c_1 and c_2 are acceleration constants. In general, c_1 is viewed as the cognitive acceleration coefficient, and c_2 is viewed as the social acceleration coefficient. A higher value of c_1 a larger deviation of the particle in the search space, while a higher value of c_2 signifies a faster convergence to the current global best position $Gbest$. The velocity update equation shown in Eq. (1) includes three parts: the first denotes the previous velocity, the second part is the cognition part used for self-exploitation, and the third part is the social part reflecting the social exploitation. The velocity of each particle is limited to within the range $[V_{min}, V_{max}]$, and the position is limited to within the range $[X_{min}, X_{max}]$ to avoid the dimensional value moving too far out of the search space. The algorithm conducts the update equation iteratively until an optimum solution is found or the predefined number of iterations is met. The process of the conventional basic PSO algorithm is described in Algorithm 1.

Algorithm 1: Basic PSO algorithm.

Input: ˜Iterator times: T; Population: M; Dimension: D.
Output: ˜Global best particle's position $\boldsymbol{x}^*(t)$

$t \leftarrow 1(initialization)$;
initialize all particles position $\boldsymbol{x}_i^j$;
initialize all particles velocity $\boldsymbol{v}_i^j$;
while $((|f(x(t)^*| \geq \varepsilon)$ *or* $(t \leq T))$ **do**
  for $i = 1$ *to* M **do**
    $f(x_i(t)) \leftarrow$ Evaluate Fitness of Particle$(x_i(t))$;
    Update the $pbest_i$ position of particle i;
    Update the $gbest_t$ position in the tth iteration;
  for $i = 1$ *to* M **do**
    for $j = 1$ *to* D **do**
      $\boldsymbol{v}_i^j(t) \leftarrow \boldsymbol{v}_i^j(t) + c_1 r_1 * (\boldsymbol{P}_i^j(t) - \boldsymbol{x}_i^j(t)) + c_2 r_2 * (\boldsymbol{G}^j(t) - \boldsymbol{x}_i^j(t))$;
      $\boldsymbol{x}_i^j(t) \leftarrow \boldsymbol{x}_i^j(t) + \boldsymbol{v}_i^j(t)$;
  $t \leftarrow t + 1$;
return $\boldsymbol{x}^*(t)$;

2.2. Review of PSO variants

Since the PSO was introduced by Kennedy and Eberhart in 1995, a substantial number of PSO variants have been proposed by researchers focusing on providing better performance and addressing problems such as particle trapping in local optima or low accuracy on complex problems. These improved variants of the PSO algorithm can roughly be classified into four categories: (a). Parameter modification based algorithms, (b). Neighborhood topology-based algorithms, (c). Learning strategies-based algorithms, and (d). Hybridized method-based algorithms.

2.2.1. Parameter modification based algorithms

In PSO update equation Eq. (1), the proper values for parameters such as the inertia weight, and acceleration coefficients have a significant influence on the convergence of a swarm. Shi and Eberhart first introduced an inertia weight ω to the PSO velocity update equation in Eq. (2).

$$V_i^D(t+1) = \omega V_i^D(t) + c_1 r_1 (Pbest_i^D(t) - X_i^D(t)) + c_2 r_2 (Gbest_i^D(t) - X_i^D(t)) \tag{2}$$

It was designed to balance the exploration and exploitation search abilities. In general, a large inertia weight is more appropriate for global search, and a small inertia weight tends to facilitate local search. Nickabadi [28] divided the various inertia weighting modifications into three groups: (1). Constant and random inertia weight, such as a fixed or random constant where no input information is required. (2). Time varying inertia weight, which is defined based on a function of time or

the number of iterations. These methods can be linear or nonlinear and increasing or decreasing, e.g., linearly increasing, nonlinear increasing, linearly decreasing, nonlinear decreasing, random and chaotic. (3). Adaptive inertia weight, which traces the search situation and adapted the inertia weight based upon feedback information of parameters, e.g., adaptive fuzzy weight, characteristic parameters based weight. A comparison study of various inertia weights in [14] showed that the linearly decreasing inertia weight is the most efficient method. Kennedy showed that the acceleration coefficients are important to the success of PSO in both the social-only model and cognitive-only model [18]. Suganthan [37] proposed ad hoc versions of c_1 and c_2, which performed better rather than a fixed value of 2.0 for different problems. The effect of the acceleration coefficient on the convergence of the PSO algorithms have studied in [3]. Moreover, mixed adaptive parameter selection by incorporating the inertia weight and acceleration coefficients was proposed in [1] . Nonuniform adjustment of parameters was proposed in [11]. In addition to inertia weight and acceleration coefficients, another modification with constriction factor was introduced into the original PSO algorithm. The constriction factor was used regularly to control the magnitudes of the particle velocities and guarantee convergence. It has been found to be very effective on some problems. The factor form is described in Eqs. (3) and (4).

$$V_i^D(t+1) = \chi \cdot \left(V_i^D(t) + c_1 r_1 (Pbest_i^D(t) - X_i^D(t)) + c_2 r_2 (Gbest^D(t) - X_i^D(t))\right) \tag{3}$$

$$\chi = \frac{2}{|2 - \varphi + \sqrt{\varphi^2 - 4\varphi}|}, \tag{4}$$

where χ is a function of c_1 and c_2, $\varphi = c_1 + c_2$ and $\varphi > 4$. c_1 and c_2 represent the cognitive and social parameters, respectively. Typically, the algorithm is often used with $\chi = 0.7298$ and $c_1 = c_2 = 2.05$. A detailed theoretical analysis of the constriction factor can be found in [10].

2.2.2. Neighborhood topology based approaches

The propose of introducing topology structure in PSO aims to enrich population diversity. Topology structure influences the flow rate of the best information between individuals. Kennedy [16] claimed that a small neighborhood of PSO tends to yield better performance on complex problems, and a large neighborhood is better for solving simple problems. Improving PSO performance by designing different neighbor topologies is an active research area. Several classical topologies such as fully connected, ring and Von Neumann were proposed in [19]. Mendes and Kennedy [27] introduced a fully informed PSO (FIPS) that adopts a weighted sum of all of the topological information of its neighbors instead of the only *Pbest* or *Gbest* information to update the position of a given particle. Instead of using a fixed neighborhood topology, a dynamic multi-swarm PSO (DMSPSO) was proposed by Suganthan and Liang [23]. The whole swarm is randomly divided into small groups with few neighbors in an early stage for a better exploration. Then, these swarms are regrouped periodically using various regrouping schedules and information is exchanged among the sub-swarms for a better exploitation in the later stages. In [26], Marinakis et al.proposed an expanding neighborhood topology (PSOENT) with variable neighborhood search strategy. In [46], Wang et al. proposed a dynamic tournament topology PSO. Many other dynamically changing neighborhood structure were introduced in [30].

2.2.3. Learning strategies based approaches

Recently, another area of research is to explore the PSO learning strategy. Many PSO variants with different learning strategies have been proposed for better convergence. Peram and Veeramachaneni [32] developed a fitness-distance-ratio based particle swarm optimization(FDRPSO) with near neighbor interactions. It moves particles towards nearby particles of higher fitness instead of towards the global best position. Liang and Suganthan [22] proposed a comprehensive learning particle swarm optimization (CLPSO) for solving complex multimodal functions. It updates the velocity of a given particle by learning from either it's own *Pbest* position or the *Pbest* position of other selected particle in each dimension. In [43], a cooperative approach to PSO (CPSO) was proposed to solve large-scale optimization by using multiple swarms to optimize each dimension of the solution vector independently and showed better performance than the conventional PSO on multi-modal problems. Another similarly algorithm named cooperatively coevolving particle swarm optimization(CCPSO) in [21] employs a random dimensions regrouping technique and adopts a novel updates strategy based on Cauchy and Gaussian distributions to sample new points in the searching space. In [40], Sun et al. introduced a quantum behaved particle swarm optimization (QPSO) that has been proven to be a global guaranteed convergence optimizer. In [49], Zhan et al. introduced an orthogonal learning strategy to guide the particles toward promising regions. Recently, a growing amount of evidence has shown that the single learning strategy based optimizer is not always competent for all problems, therefore, multiple strategies have been proposed. Hu [15] introduced an adaptive PSO with multiple adaptive methods. Wang et al. [47] proposed a self adaptive learning strategy based optimizer. Tanweer [42] proposed a self regulating particle swarm optimization (SRPSO), which incorporate the self-regulating inertia weight determined by the best particle for better exploration and self-perception on global search direction determined by the rest particle's for exploitation in the search space. In [20], a self learning PSO (SLPSO) was proposed by adopting four strategies to address various of searching spaces. Another kind of learning strategy is based on the probability model, Kennedy [17] introduced a bare bone's optimizer named BBPSO, which uses a Gaussian distribution rather than a velocity equation for particle updates. In [39], a new fitness estimation strategy-based PSO was proposed in which the number of fitness evaluations is decreased through the analysis of the relationship between particles. In [8], a competitive swam optimizer that uses a pairwise competition strategy was

proposed to solve large scale optimization problems. Emerging advancements that use multi-swarm strategies have been introduced to increase the diversity of swarms, such as multi-swarm evolutionary framework based on a feedback mechanism in [9], a multi-swarm cooperative multistage perturbation optimizer in [50] and a novel concurrent PSO algorithm designed to alleviate the premature convergence problem of PSO algorithm in [4].

2.2.4. Hybridized approaches

To improve the performance of the PSO algorithm, another active research is hybrid PSO with the advantageous properties of other search techniques. One method is to combine PSO with other evolutionary algorithms, such as genetic algorithm PSO(GAPSO), simulated annealing PSO(SAPSO), ant colony optimization PSO(PSACO) , and differential evolution PSO(DEPSO). Another hybrid PSO method has been introduced by incorporating various operators, such as the mutation operators [31], and biological inspired operators, and the aging theory inspired PSO [7]. In addition, local search methods have been integrated with PSO, such as tabu search [48], and neighborhood search [45]. Recently, other hybridized approaches, such as avoidance mechanism , diversity enhancing mechanism , Newton's laws of motion and the combination of multi-crossover with bee colony mechanism [25] have been proposed to improve the searching ability of PSO. The detailed summaries of the hybrid PSO algorithms are provided in [12]. Among the above four categories, the learning strategy and the hybridized strategy research have been gained considerable attention due to the better convergence characteristics and solutions to the global optimum. Therefore, integrated multiple strategies and designed novel learning approach provide new research directions.

3. Vector coevolving particle swarm optimization

In this section, we describe our proposed algorithm, the vector coevolving particle swarm optimization (VCPSO), in detail. First, we introduce a new technique to partition a particle's full dimensions into several segments. Second, the new designed vector operators are shown. Finally, we made an analysis of the searching behavior of VCPSO.

3.1. Vector partition

In the conventional PSO and most of its variants, particle's fitness is usually determined by the values of its full dimension on some simple problems. However, for some complex problem, a fully dimensional learning strategy may cause poor solutions because the value of one dimension or the values of some parts of the full dimensions [22]. When solving large-scale or high-dimensional optimization problems, the conventional PSO algorithm show poor solution quality and slow convergence. The obvious reason for both phenomena can be attributed to the independent dimensional learning approach in which the numerical relations in inner sub-dimensions is ignored. Therefore, we designed a new method to build the relations between the adjacent element. First, we introduced a random vector partition technique to divide the full dimensions into several small segments. Then, for each segment, we build the dimensional relations between the adjacent sub-dimensions using the novel operators. The whole process of the vector partition is described in Fig. 1. When searching a D-dimensional problem, the full dimensions of each particle are randomly partitioned into $k+1$ segments with a number of k split points. These split points indices are randomly generated by the following formula in Eq. (5) :

$$S_i = Rand(S_{i-1} + m, Dim - 1 - m) \tag{5}$$

where S_i denotes the index of current split point, S_{i-1} denotes the index of the previous split point in the full dimensions. m is the minimum interval for each segment and it is set at the beginning of the search. This formula returns an integer value S_i that is generated randomly in the range of $[S_{i-1} + m, Dim - 1 - m]$. For high dimensional optimization, this formula guarantees the validity of the random splitting. As each segment contains at least m elements, and m is a constant integer number, thus, the total number of split points satisfy $k \in [1, \lfloor D/m \rfloor]$. In addition, the partition technique can guarantee that the segment contains at least m elements. Given that the split point was randomly generated in each iteration, the full dimensions of the particle can be automatically divided into several segments by the split points. The details of the process of vector partition for the full dimensions of a particle are shown in Algorithm 2.

3.2. Vector operators

When each particle has finished the vector partitioning operation, we randomly assigned either one of the vector operators to update the value of the sub-dimensions in each segments. The process of operator assignment is shown in Fig. 2. As mentioned previously, the operators are designed to build the numerical relations between inner sub-dimensions. In general, there are three kinds of relations between the adjacent element in sub-dimension: equivalent relation, greater relation and lower relation. In the searching space, these relations can be viewed abstractly as physical motions, such as moving to a relatively higher position, moving to a relatively lower position or the combinations of them. Therefore, we design four novel operators to simulate the physical motions of the particle. These operators are the increasing operator, decreasing operator, hill operator and lake operator. The last two operators are combinations of the increasing and decreasing operators. During the evolution, the particles tend to cluster together and converge to a locally or globally optimal area; hence, the operator operations can help the locally trapped particles escape from the local optima. Both of the novel techniques, namely, randomly vector partition and random operator assignment, ensure that the evolution process is self-adjusting and initiative.

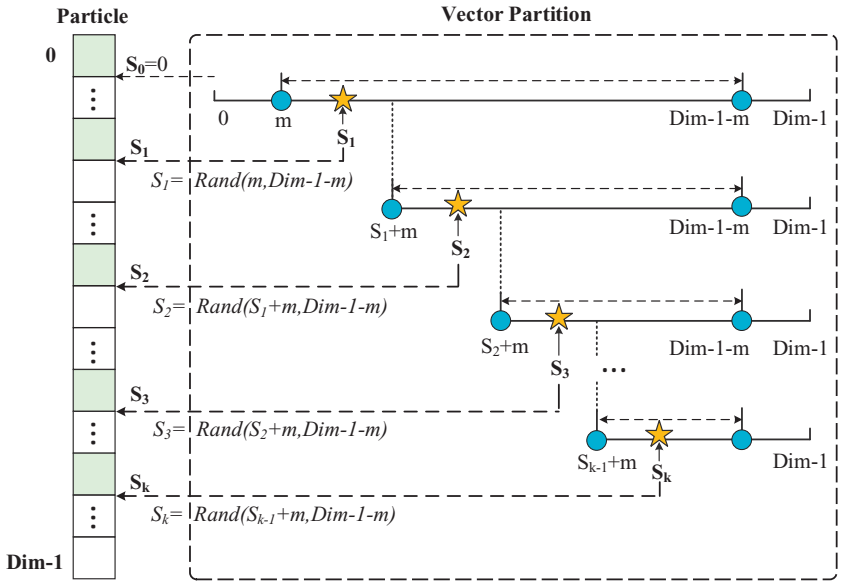


Fig. 1. Vector partition. (The star denotes a partition point, and the adjacent circles are used to indicate the scope of the selection of a partition point.

Algorithm 2: Vector Partition.

Input: Particle index: i; Dimension: D.
Output: The vector partition array : $\mathbf{S}_i[D]$.

```
Initialization: k_i ← 1;
for j = 0 to D do
    S_i[j] ← 0;
Partition:
for j = 1 to D do
    Start ← S_i[j − 1] + m;
    if (Start ≥ D − 1 − m) then
        break ;
    else
        end = Rand(Start, D − 1 − m);
        S_i[j] ← end;
        k_i ← k_i + 1;
return S_i;
```

3.2.1. Increasing operator

The increasing operator is used to build a numerical greater relation between adjacent dimension in the same segment, e.g., for jth dimension of particle i, if the value of x_i^j is lower than the value of previous dimension x_i^{j-1}, then the value of the jth dimension will be increased by Eq. (6), where F is a random number generated from a uniform distribution U(0,1), s and e are the start and end indices of the corresponding segment. In high dimensional space, a particle can be moved to a relatively higher position driven by the increasing operator operation. Clerc and Kennedy provided a theoretical analysis of particle trajectories showing that a simple particle with inertia converges to a stable point [10], which satisfied the equation $\lim_{t\to\infty} X_i^t = E\left(O_i^t\right)$, where E is an expectation function and O_i^t is an average stable point. However, if the stable point is not the global optimum position, the searching process will convergence to a local optimum. The increasing operator can help particles escape from oscillation around a local optimum. The increasing operator can be formulated in Eq. (6), where Φ is a random number generated from a uniform distribution U(0,1).

$$x_i^j(t+1) = x_i^j(t) + \Phi \cdot \left(x_i^{j-1}(t) - x_i^j(t)\right), j \in [s, e) \tag{6}$$

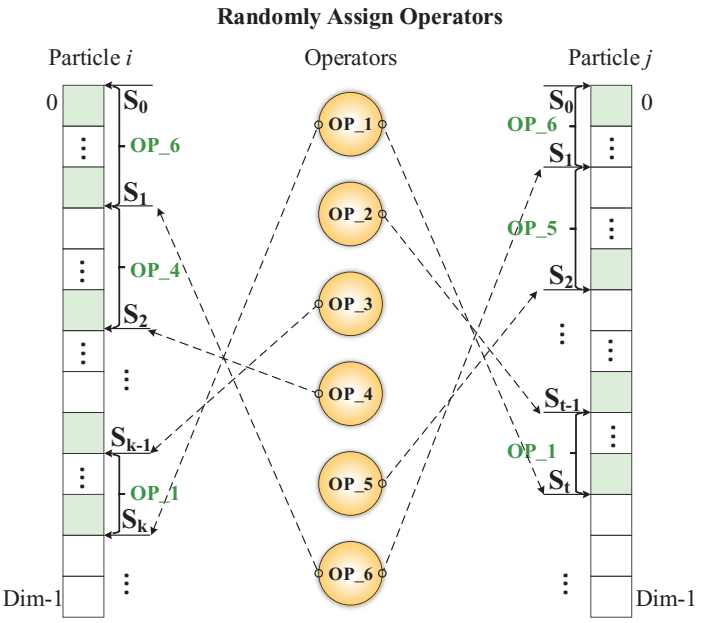


Fig. 2. Operators assignment; (OP_1∼ OP_6 denote the six operators. Each segment of a particle is assigned randomly with one of the operators.).

3.2.2. Decreasing operator

Converse to the increasing operator, the decreasing operator is used to build a lower relation between adjacent elements in a segment. In high dimensional space, a particle can move to a relatively lower position by the decreasing operator. For jth dimension of particle i, if the value of x_i^j is greater than the value of previous dimension x_i^{j-1}, then the value of the jth dimension will be decreased by Eq. (7), where Φ is a random number generated from a uniform distribution U(0,1), s and e are the start and end indices of the corresponding segment.

$$x_i^j(t+1) = x_i^j(t) - \Phi \cdot \left(x_i^j(t) - x_i^{j-1}(t)\right),\ j \in [s, e) \tag{7}$$

3.2.3. Hill operator and lake operator

The hill operator and lake operator are both composition operators. They are designed to build hybrid relationships with the dual features of the increasing and decreasing operators for adjacent dimension in a segment. First, a division point is randomly generated in a segment in the range of [s, e], where s is the start split point's index and e is the end split point's index in the segment. Then, the division point is used to divide the segment into two parts.

For the hill operator, the previous part was assigned with the increasing operator and the latter part assigned with the decreasing operator. Conversely, for the lake operator, the previous part was assigned with the decreasing operator and the latter part with the increasing operator. When one of the combined operator are assigned to one segment, they can simultaneously move the particle to another position with a few steps.

3.2.4. Centralized operator

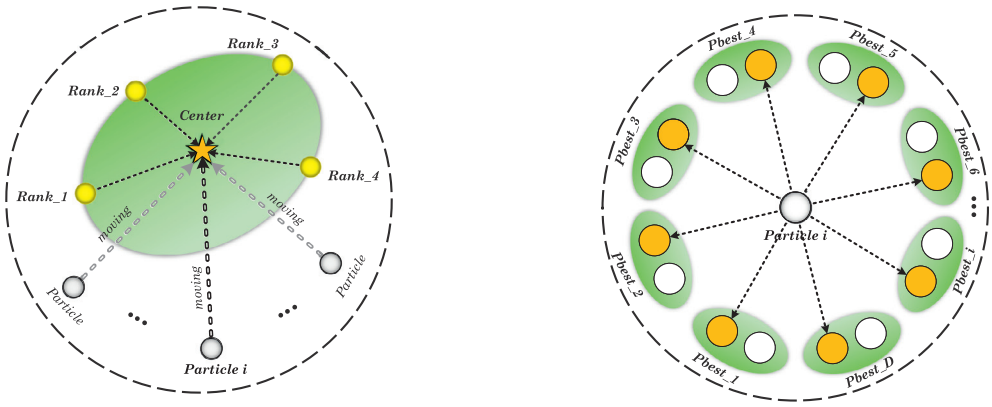
The centralized operator is designed to enhance the expatriation search ability and accelerate the convergence of VCPSO. The mean position of the number of the top k best individuals is utilized as the promising center. This operator can cluster the swarm to the global best position special for solving the unimodal problems. The number k is taken to be a minimum of two. As the fitness improves over iterations, the k-th number of the best members keep changing, in other words, dynamic.

Fig. 3-(a) shows an example of mean best position by four best individuals ($k = 4$). The star denotes the mean best ($MeanGbest_i$) position of the top k best individuals. The update formula of this operator is depicted in Eq. (8), where $\omega = 0.721$, $c = 2.00$, and r is a random number distributed uniformly in the interval (0,1).

$$\begin{cases} v_i^j(t+1) = \omega v_i^j(t) + c * r * (Center_i^j(t) - x_i^j(t)), \\ x_i^j(t+1) = x_i^j(t) + v_i^j(t+1). \end{cases} \tag{8}$$

3.2.5. Decentralized operator

The decentralized operator is designed to enhance the exploration search ability. Converse to the centralized operator, the decentralized operator updated the jth sub-dimension of particle i by learning from a stochastic particle $\gamma_i(j)$, whose *pbest* fitness performs better than another randomly selected particle. This operator can help particles avoiding swarm



(a) Centralized operator (b) Decentralized operator

Fig. 3. Schematic representation of learning operators: (a) Centralized operator (The star denotes the mean position of top ranked particles marked by Rank_i). (b) Decentralized operator (The shadowed circle denotes the learning individual which performs better than another randomly selected particle.)

clustered in the local optima especially when solving the multi-modal problems. Fig. 3-(b) shows the structure of this operator. The update formula of this operator is depicted in Eq. (9), where $\omega = 0.6$, $c = 1.0$, r is a random number distributed uniformly in the interval (0,1).

$$\begin{cases} v_i^j(t+1) = \omega v_i^j(t) + c * r * (Pbest^j_{\gamma_i(j)}(t) - x_i^j(t)), \\ x_i^j(t+1) = x_i^j(t) + v_i^j(t+1). \end{cases} \tag{9}$$

3.3. Searching behavior

A completive VCPSO algorithm is presented in Algorithm 3. The VCPSO searching process can be divided into two stages: an exploration stage and an exploitation stage. Both of the exploration and exploitation phases interact alternately during the space searching process. In the exploration phase, the population diversity has a substantial influence on the global searching, especially when some particles are trapped into optima. For the ith particle at the tth iteration in VCPSO, its full dimension was partitioned into $k_{t,\,i}$ segments and each segment was assigned with one of the operators randomly. Thus, the number of potential operator combinations value Δ_i for the ith particle at the tth iteration satisfies the equations as $\Delta_i = N^{k_{t,i}}$, where N is the number of the operators, $k_{t,\,i}$ denotes the number of partitioned segments in particle i at the t-th iteration. The total number of potential operator operations by the M particles through the T iterations satisfies the equation $\Delta = \sum_{t=1}^{T}(N^{k_{t,1}} + N^{k_{t,2}} + \cdots + N^{k_{t,i}} + \cdots + N^{k_{t,M}})$. Each combination of the different operators represents one solution-update strategy in the search space, so these operator operations can bring more candidate solutions which can enrich the diversity of the population and can help particles escape from local optimum. In the exploitation phase, the global centralized operator utilized the mean position of the rank topped neighbors to reserve the global optimal information and to guide the particles flying to the global optimum. While the local decentralized operator learning from anyone potential particle can reserve more local optimal information of current swarm. The two information-based learning operators can guarantee that particles move toward promising areas near the global optimum. The exploration and exploitation phases interact alternately during the space searching process.

4. Experimental data, results and discussion

In this section, we focused on the experiment to evaluate the performance of VCPSO and some PSO variants based on the benchmark functions. The content of this section is organized as follows: A detailed description of the general used benchmark functions was given firstly. Then, the impact of the designed vector partition technique and the update operators in VCPSO algorithm were evaluated. Next, the performance of VCPSO and other algorithms were evaluated on these test problems. Finally, a convergence analysis, statistical comparison analysis and the computational complexity are provided.

4.1. Tested benchmark functions

The test functions contains twenty-five benchmark functions which are widely used for evaluating the performance of the evolution optimization algorithms. The functions from f_1 to f_{16} are from [2,22] and the function f_{17} to f_{25} are selected from the special session on real-parameter optimization in [38]. These tested functions in the study are classified into four parts: unimodal functions, multi-modal functions, rotated functions and shifted functions. The properties and the formulas of these benchmark functions are shown in Table 1.

Algorithm 3: VCPSO: vector coevolving particle swarm optimization algorithm.

Input: Iterator times: T; Population: M; Dimension: D.
Output: The global best particle's position $\boldsymbol{x}^*(t)$

$t \leftarrow 1(initialization)$;
for $i = 1$ *to* M **do**
  for $j = 1$ *to* D **do**
    initialize each particle i's jth position x_i^j;
    initialize each particle i's jth velocity v_i^j;
  Vector Partition by **Algorithm** 2;
  Assign Operators for each segment;
while $(|f(\boldsymbol{x}(t)^*| \geq \varepsilon)$ *or* $(t \leq T)$ **do**
  for $i = 1$ *to* M **do**
    $f(x_i(t)) \leftarrow$ Evaluate Fitness of Particle$(x_i(t))$;
    Update the $pbest_i$ position of the particle i;
    Update the $gbest_t$ position in the tth iteration;
  for $i = 1$ *to* M **do**
    if $(PbestNoUpdateTimes_i > 2)$ **then**
      (a). *Vector Partition* by **Algorithm** 2;
      (b). *Assign operator* for each segment;
      (c). $PbestNoUpdateTimes_i \leftarrow 0$;
    for $k = 1$ *to* k_i **do**
      /* k_i denotes the parititioned segements in particle i */
      Update particle i's $\boldsymbol{X}_i$ and $\boldsymbol{V}_i$ with corresponding operators;
  $t \leftarrow t + 1$;
return the best position $\boldsymbol{x}^*(t)$;

The functions $F_1 - F_6$ are unimodal functions. $F_1 - F_3$ are basic unimodal function. F_4 is a Rosenbrock function, it is unimodal in a 2-D or 3-D search space. As there is a narrow valley between the perceived local optima and the global optimum, it can be viewed as a multi-modal function in high dimensional space [34]. F_5 is a step function, which is characterized by plateaus with discontinuities; it has finitely many pieces. F_6 is a polynomial function with random noise. VCPSO performs significantly better than other comparable algorithms on this unimodal function, especially with high dimensions.

The functions from F_7 to F_{12} are multimodal functions. Ackleys function F_7 has only one narrow global optimum and many minor local optima. It is widely used as the multimodal test function and most likely the easiest problem among this group because its local optima are shallow. Griewangk's function F_8 has a $\prod_{i=1}^{D} cosx_i/\sqrt{i}$ component leading to linkages among variables. It is similar to Rastrigin's function and has many widespread local regularly distributed minima. Therefore, the search algorithms tend to converge in the wrong direction. Rastrigin's function F_9 is a complex multimodal problem with a large number of local minima. Weierstrass function is a pathological example of real-valued function on the real line. It has the distinct property of being continuous everywhere but difference nowhere. All of the these minimal positions are regularly distributed. When solving this function, algorithms may easily fall into local optima; therefore, an algorithm that maintains a larger population diversity is more likely to generated better solution. F_{11} is a generalized penalized function. When the searching result is far from the global optimum, the search algorithms are prone to converge in the wrong direction with bad fitness.

The functions from F_{12} to F_{15} are rotated multi-modal functions based on the multimodal function's $\boldsymbol{x}$ by left multiplication with an orthogonal matrix M to get the new rotated variable $\mathbf{y} = M \cdot \boldsymbol{x}$, as follows in Eq. (10).

$$\boldsymbol{y} = \underbrace{\begin{bmatrix} a_{11} & a_{11} & \dots & a_{1D} \\ a_{21} & \dots & \dots & a_{2D} \\ \dots & \dots & \dots & \dots \\ a_{D1} & a_{D2} & \dots & a_{DD} \end{bmatrix}}_{M} \cdot \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}}_{X} \tag{10}$$

where the variable $\boldsymbol{y}$ is used to calculated the function's value. $[x_1, x_2, \cdots, x_D]^T$ is a D-dimension vector, then $y_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{iD}x_D, i = 1, 2, \dots, D,$ when some of the values in vector $\boldsymbol{x}$ are changed, all dimensions in $\boldsymbol{y}$ will change as well. Hence, the rotated function cannot be solved using one-dimensional searches. The orthogonal rotation matrix M does not affect the shape of the functions.

Table 1
The properties and the formulas of the test benchmark functions ($F_1 \sim F_{28}$).

Name	Objective	Search Range	Global minimum	Modality
Sphere function	$F_1 = \sum_{i=1}^{D} x_i^2$	[-100,100]	0	Unimodal
Schwefel's function P1.2	$F_2 = \sum_{i=1}^{D} \sum_{j=1}^{D} x_j^2$	[-100,10]	0	Unimodal
Schwefel function P2.22	$F_3 = \sum_{i=1}^{D} \|x_i\| + \prod_{i=1}^{D} \|x_i\|$	[-10,10]	0	Unimodal
Rosenbrock function	$F_4 = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	[-10,10]	0	Unimodal
Step function	$F_5 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	[-10,10]	0	Unimodal
Noise Quartic function	$F_6 = \sum_{i=1}^{D} i x_i^4 + random[0, 1)$	[-1.28,1.28]	0	Unimodal
Ackley's function	$F_7 = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i))) + 20 + e$	[-32,32]	0	Multimodal
Griewank function	$F_8 = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0	Multimodal
Rastrigin's function	$F_9 = \sum_{i=1}^{D} (x_i^2 - 10cos(2\pi x_i) + 10)$	[-5.12,5.12]	0	Multimodal
Weierstrass function	$F_{10} = \sum_{i=1}^{D} (\sum_{k=0}^{20} [0.5^k cos(2\pi * 3^k (x_i + 0.5))]) - D\sum_{k=0}^{20} [0.5^k cos(2\pi * 3^k * 0.5)]$	[-0.5,0.5]	0	Multimodal
Generalized Penalized function	$F_{11} = \frac{\pi}{D}\{10\sin^2(\pi y_1) + A + (x_D - 1)^2\} + B$ $A = \sum_{i=1}^{D-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})]$, $B = \sum_{i=1}^{D} u(x_i, 5, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a, \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m. & x_i < -a. \end{cases}$	[-50,50]	0	Multimodal
Rotated Ackley's function	$F_{12} = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi y_i))) + 20 + e, \boldsymbol{y} = \boldsymbol{M} \cdot \boldsymbol{x}$	[-32,32]	0	Multimodal
Rotated Griewank's function	$F_{13} = \sum_{i=1}^{D} \frac{y_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{y_i}{\sqrt{i}}) + 1, \boldsymbol{y} = \boldsymbol{M} \cdot \boldsymbol{x}$	[-600,600]	0	Multimodal
Rotated Rastrigin's function	$F_{14} = \sum_{i=1}^{D} (y_i^2 - 10cos(2\pi y_i) + 10), \boldsymbol{y} = \boldsymbol{M} \cdot \boldsymbol{x}$	[-5.12,5.12]	0	Multimodal
Rotated Noncontinuous Rastrigin's function	$F_{15} = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10), z_i = \begin{cases} y_i & \|y_i\| < 0.5, \\ \frac{round(2y_i)}{2} & \|y_i\| \ge 0.5 \end{cases}$	[-5.12,5.12]	0	Multimodal
shifted sphere	$F_{16} = \sum_{i=1}^{D} z_i^2 + f_b, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$	[-100,100]	−450	Unimodal
Shifted Schwefels problem 1.2	$F_{17} = \sum_{i=1}^{D} (\sum_{j=1}^{D} z_j^2) + f_b, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$	[-100,100]	−450	Unimodal
Shifted rotated high conditioned elliptic	$F_{18} = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_b, \boldsymbol{z} = \boldsymbol{M} \cdot (\boldsymbol{x} - \boldsymbol{o})$	[-100,100]	−450	Unimodal
Schwefels problem 2.6 with global optimum on bounds	$F_{19} = max\|A_i - B_i\| + f_b$,	[-100,100]	−310	Unimodal
shifted Rosenbrock's function	$F_{20} = \sum_{i=1}^{D} (100(z_i^2 - z_i + 1)^2 + (z_i - 1)^2 + f_b, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$	[-100,100]	390	Multimodal
Shifted rotated Griewanks function without bounds	$F_{21} = \sum_{i=1}^{D} (\sum_{j=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{z_i}{\sqrt{i}}) + 1 + f_b, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$	[0,600]	−180	Multimodal
Shifted rotated Ackleys function with global optimum on bounds	$F_{22} = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi z_i))) + 20 + e + f_b, \boldsymbol{z} = \boldsymbol{M} \cdot (\boldsymbol{x} - \boldsymbol{o})$	[-32,32]	−140	Multimodal
Shifted Rastrigins Function	$F_{23} = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + f_b, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$	[-5,5]	−330	Multimodal
Shifted rotated Rastrigins function	$F_{24} = \sum_{i=1}^{D} (z_i^2 - 10cos(2\pi z_i) + 10) + f_b, \boldsymbol{z} = \boldsymbol{M} \cdot (\boldsymbol{x} - \boldsymbol{o})$	[-100,100]	−330	
Shifted rotated Weierstrass function	$F_{25} = \sum_{i=1}^{D} (\sum_{k=0}^{kmax} [a^k cos(2\pi * b^k (z_i + 0.5))]) - D\sum_{k=0}^{kmax} [a^k cos(2\pi * b^k * 0.5)], \boldsymbol{z} = \boldsymbol{M} \cdot (\boldsymbol{x} - \boldsymbol{o})$	[-100,100]	90	Multimodal
Schwefel's function 2.13	$F_{26} = \sum_{i=1}^{D} (\|A_i - B_i(x)\|)^2 + f_b$, $A_i = \sum_{j=1}^{D} (a_{ij} sin\alpha_j + b_{ij} cos\alpha_j)$, $B_i(x) = \sum_{j=1}^{D} (a_{ij} sinx_j + b_{ij} cosx_j)$	[-100,10]	−460	Multimodal
F_{27}:Shifted expanded Griewanks plus Rosenbrocks function ($G_{(x)} : F_{21}, R_{(x)} : F_{20}$)	$F_{27} = G(R(z_1, z2)) + G(R(z_2, z3)) + \cdots + G(R(z_D, z_1)) + f_b, \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$	[-5,5]	−130	Multimodal
Shifted rotated expanded Scaffer's F_6 function	$F_{28} = F(z_1, z_2) + F(z_2, z_3) + \cdots + F(z_{D-1}, z_D) + f_b$, $F(x, y) = 0.5 + \frac{sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.0001(x^2+y^2))^2}$, $\boldsymbol{z} = \boldsymbol{M} \cdot (\boldsymbol{x} - \boldsymbol{o})$	[-5,5]	−300	Multimodal

The functions from f_{16} to f_{28} are shifted functions. The shifted global optimum for all test functions is $\boldsymbol{o} = [o_1, o_2, \ldots, o_D]$. The function defined as $\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})$ is used for shifted functions and $\boldsymbol{z} = \boldsymbol{M} \cdot (\boldsymbol{x} - \boldsymbol{o})$ is used for shifted rotated function, where the $\boldsymbol{M}$ is the transformation matrix for the rotating matrices. Some of the shifted or roted benchmark functions in are showed in Fig. 4.

4.2. Study the effects of the techniques in VCPSO

VCPSO contain six operators, namely: increasing operator, decreasing operator, hill operator, lake operator, centralized operator and decentralized operator. The former four operators operators can be viewed as the scalar operators as they update the position of a given particle depended on the numerical relationships between its inner dimensions without using the velocity $\boldsymbol{v}$ to guide its optimization. While the latter two operators can be considered as the vector operators, as they utilize the velocity $\boldsymbol{v}$ to update a particle's position. The effect of these operators in terms of convergence speed

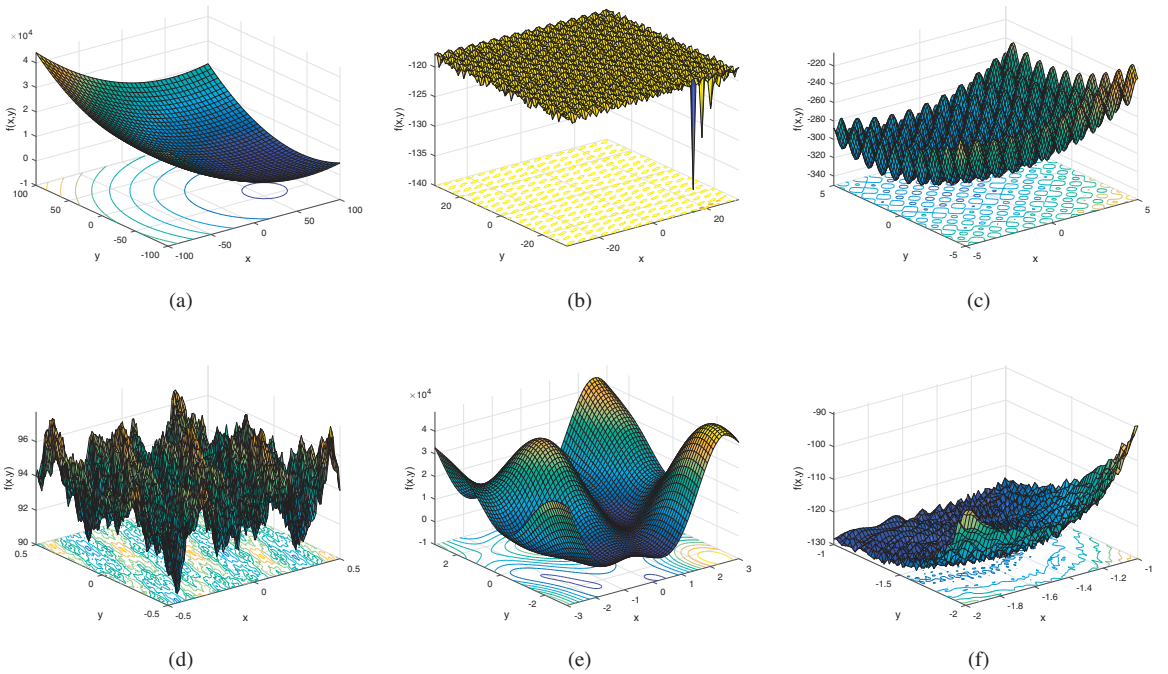


Fig. 4. The 3-D maps for 2-D functions: (a) Shifted sphere function. (b) Shifted rotated Ackley's function with global optimum on bounds. (c) Shifted Rastrigin's function. (d) Shifted rotated Weierstrass function. (e) Schwefel's problem 2.13. (f) Shifted expanded Griewank's plus Rosenbrock's function.

and convergence accuracy are studied. The experimental tests in the following are run twenty-five times independently on the selected different types of functions on 10-D and 30-D problems.

4.2.1. Analysis and effects of the operators on convergence.

Table 2 compared the average number of fitness evaluations (FEs) of the scalar operators and the hybrid learning version before reaching the fixed accuracy level on selected benchmark functions for the 10-D, 30-D problems. The scalar operators without the learning strategy is marked with *Scalar Operators* and the hybrid version is marked with *Hybrid Operators*. F_1 and F_4 were selected representative the unimodal problems, F_7 and F_8 representative the multimodal problems, and F_{14} and F_{15} representative the rotated multimodal problems. The mean fitness evaluation times(*FEs*), standard deviation (*StD.*), success rate (*SR*) are showed in Table. 2. The better FEs are shown in bold. It is obviously that the hybrid operators convergence to the fixed accuracy with higher success rate and lower evaluations than the scalar operators on almost all of the benchmark functions. The experimental results demonstrated the effectiveness and feasibility of the hybrid strategy in VCPSO.

4.2.2. Analysis and effects of the operators on accuracy

In VCPSO, six operators are introduced to optimize the sub-dimensions of a particle separately. Each of the operator performed its own search behavior. The independent search performance and the co-evolving search performance are tested on 10-D, 30-D, and 50-D problems on test benchmark functions from F_1 to F_{15}. In order to make a fair comparison among operators in VCPSO, each test shared the same parameter settings, such as the population size, fitness evaluations, et.al. and the default minimum partitioning interval m for each segment is set to two. The accuracy of the mean best fitness are chosen as the measure item. Comparison results are provided in Table. 3. The mean best fitness values (*Mean*) are showed in bold.

From the results in Table 3, one can see that the overall hybrid operators can obtain the highest convergence accuracy on almost all of the test functions on average, specially for the complex unimodal problems, as Rosenbrock's function F_4, multimodal problems, as Rastrigin's function F_9 and the rotated multimodal functions, as F_{12}, F_{13}, F_{14} and F_{15}. In addition, the co-evolving of the centralized with decentralized operator also show better performance than the independent version on complex problems and can be comparable with the hybrid strategy. This property that the hybrid version in solving most of the complex unimodal and multimodal problems is due to the VCPSO's co-evolving strategy that inherit the advantage of the independent operator.

4.2.3. Analysis and effects of the vector partition

In VCPSO, the vector partition technique is designed to partition the full dimension of each particle into the number of k segments. For each segment, it contains at least the number of m elements and these elements evolved in one segment can be optimized by one of the operators independently. The minimum interval size m directly determines the size of k. The range of k satisfies $k \in [1, \lfloor D/m \rfloor]$. For example, for a 10 dimensional problem, if $m = 3$, the full dimension of a particle

Table 2
Comparison of the average number of fitness evaluations of different operators before reaching the fixed accuracy level.

Func.	Scalar operators(four operators)				Hybrid operators (six operators)			
	Mean	*StD.*	SR (%)	FES	*Mean*	*StD.*	SR(%)	FES
10-D problems								
F_1	7.80E−07	4.61E−07	100%	1.90E+04	7.10E−07	8.05E−07	100%	**•1.52E+04**
F_2	7.22E−07	5.81E−07	100%	2.03E+04	7.47E−07	4.18E−07	100%	**•1.65E+04**
F_3	8.68E−07	9.57E−07	100%	2.82E+04	8.41E−07	3.08E−07	100%	**•2.31E+04**
F_4	6.73E−07	2.69E−09	100%	**•1.77E+04**	7.11E−07	9.37E−07	100%	2.92E+04
F_5	0.00E+00	0.00E+00	100%	7.31E+03	0.00E+00	0.00E+00	100%	**•5.60E+03**
F_6	7.04E−03	1.11E−02	100%	1.85E+04	6.53E−03	7.73E−03	100%	**•1.74E+04**
F_7	8.20E−07	7.33E−08	100%	2.72E+04	8.45E−07	4.99E−08	100%	**•2.31E+04**
F_8	9.40E−03	4.20E−02	80%	7.89E+04	7.26E−07	7.25E−07	100%	**•6.50E+04**
F_9	7.61E−07	6.32E−07	100%	5.43E+04	7.80E−07	1.60E−07	100%	**•4.29E+04**
F_{10}	6.90E−07	6.15E−07	100%	3.77E+04	7.11E−07	2.10E−07	100%	**•3.38E+04**
F_{11}	6.02E−07	1.04E−06	100%	1.34E+04	7.03E−07	2.77E−07	100%	**•1.08E+04**
F_{12}	8.79E−07	8.43E−07	100%	2.74E+04	7.86E−07	2.58E−07	100%	**•2.31E+04**
F_{13}	7.53E−07	9.91E−07	100%	3.22E+04	7.91E−07	6.27E−07	100%	**•3.21E+04**
F_{14}	6.61E−07	1.97E−07	100%	3.55E+04	6.99E−07	8.75E−07	100%	**•3.02E+04**
F_{15}	6.85E−07	4.45E−07	100%	3.59E+04	7.05E−07	4.82E−07	100%	**•3.20E+04**
30−D problems								
F_1	8.56E−07	8.46E−08	100%	5.05E+04	8.19E−07	1.82E−07	100%	**•2.99E+04**
F_2	8.34E−07	2.89E−07	100%	5.66E+04	8.57E−07	2.43E−07	100%	**•3.38E+04**
F_3	9.26E−07	2.58E−07	100%	8.43E+04	9.27E−07	9.27E−07	100%	**•4.89E+04**
F_4	7.36E−07	5.42E−07	100%	**•4.19E+04**	8.81E−07	4.31E−07	100%	6.74E+04
F_5	0.00E+00	0.00E+00	100%	2.06E+04	0.00E+00	0.00E+00	100%	**•1.13E+04**
F_6	7.56E−03	4.99E−04	100%	1.80E+05	7.51E−03	2.10E−03	100%	**•1.22E+05**
F_7	9.34E−07	3.01E−08	100%	6.90E+04	9.28E−07	1.96E−07	100%	**•4.29E+04**
F_8	4.59E−02	2.05E−01	85%	1.94E+05	8.71E−07	1.21E−06	100%	**•8.96E+04**
F_9	1.20E+01	1.23E+01	40%	2.83E+05	8.33E−07	1.85E−07	100%	**•1.06E+05**
F_{10}	7.90E−07	2.99E−07	100%	1.39E+05	8.64E−07	5.92E−07	100%	**8.03E+04**
F_{11}	7.66E−07	1.00E−06	100%	3.52E+04	8.56E−07	5.65E−07	100%	**•2.04E+04**
F_{12}	9.26E−07	4.30E−08	100%	6.85E+04	9.17E−07	1.82E−08	100%	**•4.31E+04**
F_{13}	8.23E−07	1.29E−06	100%	6.34E+04	8.43E−07	5.46E−07	100%	**•3.30E+04**
F_{14}	4.67E+00	3.97E+02	95%	1.93E+05	8.52E−07	6.19E−07	100%	**•5.94E+04**
F_{15}	3.39E+00	1.52E+01	100%	2.01E+05	8.82E−07	1.83E−07	100%	**•5.65E+04**

can be partitioned at most of $\lfloor 10/3 \rfloor$, namely 3 segments. When the size of m is set at a large value approximated to the particle's dimension D, the particle's dimension will be partitioned into only one segments, meanwhile, assigned with one operator. This extreme case may degenerate the capability of operator's co-evolving. Therefore, the value of m should be made a trade-off analysis. As the maximum of k is equal to $\lfloor D/m \rfloor$, therefore, for 10-D problems, the minimum interval size is evaluated from $m = 1$ $(k = 10)$ to $m = 6$ $(k = 1)$, and $m = 1$ $(k = 30)$ to $m = 16$ $(k = 1)$ for 30-D problems. F_1, F_3, F_4 were selected representative unimodal problems, F_7, F_8, F_9 were selected representative multimodal problems, and F_{13}, F_{14}, F_{15} were selected representative rotated multimodal problems. Fig. 5 presents the test results of the mean best fitness accuracy with different m on 10-D problems. From the curves on most function cases, it can find that different settings of m can bring different level of accuracy. There is not an exact optimal setting of m that can bring the best accuracy for each function. However, when the value of m is set nearby $m = 3$ or $m = 4$, VCPSO can acquire a higher accuracy on most functions than other settings in average. We also test the effects of m on 30-D problems. Fig. 6 presents the test results. It showed the same characteristics as the 10-D problems. From the curves of each function, we can find that when the value of m locate at the range from 9 to 12, most functions could generate a higher accuracy. We also tested the different dimensions of 50-D, 100-D etc., numerical experiments showed that when the value of m is set in the range of $[\lfloor 0.2^*D \rfloor, \lfloor 0.4^*D \rfloor]$, VCPSO converges faster and achieves a better accuracy on average.

4.3. Comparison of VCPSO with PSO variants

In this subsection, we will test the performance capabilities of the VCPSO algorithm based on twenty-eight benchmark functions listed in Table 1. These functions are classified into four categories, as unimodal function, multimodal function, rotated function and the shifted function. For a comparative analysis, the widely accepted PSO variants by the evolutionary computing research community are selected as the comparison algorithms.

4.3.1. Parameter settings for the compared PSO variants

Table 4 describes the parameter settings for all algorithms. All of the parameter settings are based on the suggestions in the corresponding original references.

The first two are the traditional Global version PSO(GPSO) [35] and Local version PSO(LPSO) [19]. The third is the Standard PSO (SPSO) proposed by Kennedy [6]. It summarized the experiences of various PSO variants and suggested as a stan-

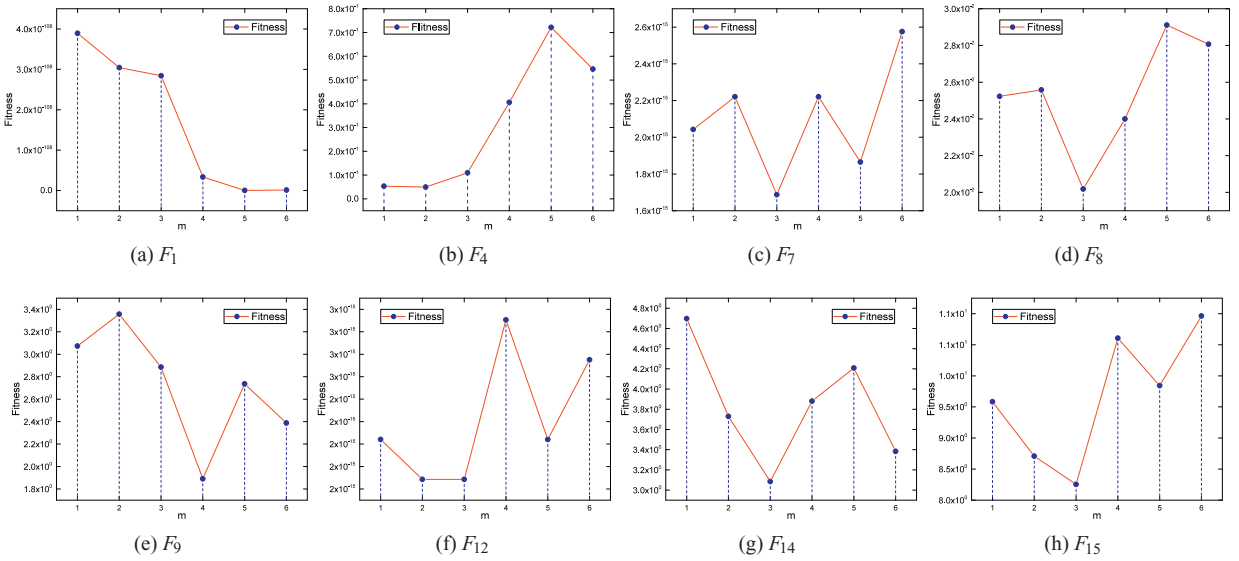
Table 3
Comparison of the mean convergence accuracy of different operators on 15 benchmark functions.

Func.	Centralized operator		Decentralized operator		Centralized-decentralized		Hybrid operators	
	Mean	*StD.*	*Mean*	*StD.*	*Mean*	*StD.*	*Mean*	*StD.*
10-D								
F_1	**8.76E−131**	1.58E−130	7.45E−129	1.60E−128	2.86E−108	5.03E−108	3.74E−85	1.43E−84
F_2	3.38E−128	7.52E−128	**7.63E−129**	8.38E−129	3.83E−108	4.76E−108	3.05E−84	9.04E−84
F_3	2.51E−65	7.73E−67	**1.11E−65**	2.50E−66	1.10E−54	6.64E−55	6.66E−44	4.48E−43
F_4	1.25E+00	1.98E+00	4.09E+00	2.82E+00	1.03E−02	5.45E−04	**0.00E+00**	0.00E+00
F_5	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F_6	5.67E−02	7.18E−02	2.26E−03	8.18E−04	1.73E−03	1.33E−03	**5.62E−04**	1.03E−03
F_7	6.13E−15	1.91E−14	2.58E−15	4.77E−15	2.58E−15	3.18E−15	**4.44E−16**	0.00E+00
F_8	4.02E−01	1.37E−01	**1.30E−02**	2.91E−02	2.36E−02	1.88E−02	**0.00E+00**	0.00E+00
F_9	3.00E+01	7.34E+01	3.19E+00	7.13E+00	2.39E+00	3.56E+00	**1.04E+00**	9.35E+00
F_{10}	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F_{11}	**4.71E−32**	1.22E−47	**4.71E−32**	1.22E−47	**4.71E−32**	1.22E−47	**4.71E−32**	1.22E−47
F_{12}	4.00E−15	0.00E+00	3.29E−15	1.59E−15	2.58E−15	4.77E−15	**4.44E−16**	0.00E+00
F_{13}	3.20E−01	6.78E−01	5.74E−02	7.32E−02	3.20E−02	7.70E−02	**0.00E+00**	0.00E+00
F_{14}	2.39E+01	4.81E+01	1.80E+01	2.27E+01	2.59E+00	1.33E+00	**0.00E+00**	0.00E+00
F_{15}	2.52E+01	1.67E+00	1.76E+01	3.47E+00	1.03E+01	7.84E+00	**0.00E+00**	0.00E+00
30-D								
F_1	**1.65E−275**	0.00E+00	4.67E−161	9.14E−161	6.46E−165	0.00E+00	5.61E−141	5.86E−141
F_2	**2.63E−273**	0.00E+00	2.52E−159	6.70E−159	1.11E−162	0.00E+00	2.39E−140	3.51E−140
F_3	2.91E−05	6.50E−05	**6.41E−82**	6.57E−82	9.96E−82	3.25E−81	1.33E−71	2.16E−71
F_4	2.16E+01	3.05E+00	1.78E+01	6.12E−01	1.36E+01	1.16E+00	**0.00E+00**	0.00E+00
F_5	4.00E+00	4.47E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
F_6	9.72E−01	1.83E+00	1.77E−02	4.61E−03	2.07E−02	4.16E−03	**3.42E−03**	2.12E−03
F_7	1.86E−01	4.16E−01	4.00E−15	0.00E+00	4.00E−15	0.00E+00	**4.44E−16**	0.00E+00
F_8	1.55E−16	1.49E−16	0.00E+00	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
F_9	1.96E+02	2.69E+02	1.34E+02	1.26E+01	1.57E+01	4.89E+00	**0.00E+00**	0.00E+00
F_{10}	2.01E+00	7.09E−02	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F_{11}	1.57E−32	0.00E+00	1.57E−32	0.00E+00	**1.57E−32**	0.00E+00	**1.57E−32**	0.00E+00
F_{12}	3.53E−14	3.81E−14	4.00E−15	0.00E+00	4.00E−15	0.00E+00	**2.58E−15**	3.18E−15
F_{13}	1.38E−02	1.42E−02	1.11E−16	0.00E+00	**0.00E+00**	0.00E+00	**0.00E+00**	0.00E+00
F_{14}	1.13E+02	2.35E+02	1.53E+02	3.64E+01	2.11E+01	6.23E+00	**0.00E+00**	0.00E+00
F_{15}	2.01E+02	2.99E+01	1.48E+02	1.61E+01	3.94E+01	1.15E+01	**0.00E+00**	0.00E+00

Table 4
Parameters settings for involved algorithms.

Algorithms	Parameter settings	Reference
GPSO	$\omega = 0.9 \sim 0.4, c_1 = c_2 = 1.193$	[35]
LPSO	$\omega = 0.9 \sim 0.4, c_1 = c_2 = 2.0$	[19]
SPSO	$\omega = 0.721,\ c_1 = c_2 = 1.193, K = 3$	[6]
QPSO	$\alpha = 1.0 \sim 0.5, u \in [0, 1]$	[41]
CLPSO	$\omega = 0.9 \sim 0.4, c = 1.49445, m = 7,$	[22]
FIPS	$\chi = 0.729, \sum c_i = 4.1, Neighbour = 2$	[27]
DMSPSO	$\omega = 0.9 \sim 0.2, c_1 = c_2 = 2.0, m = 3, R = 5$	[23]
FDRPSO	$\omega = 0.9 \sim 0.4, \phi = \{1, 1, 2\}$	[32]
VCPSO	$\omega_1 = 0.721, c_1 = 2.0,\ \omega_2 = 0.60, c_2 = 1.0,\ m = 2$	

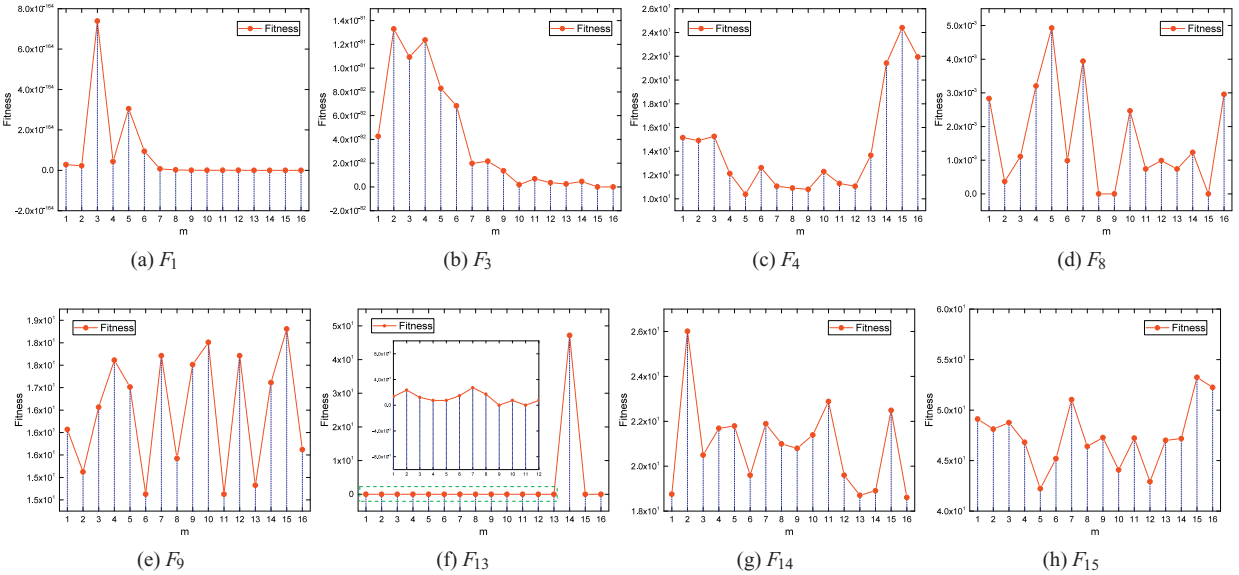
dard algorithm of evaluating the performance of the improved PSO algorithm. The fourth is Quantum PSO(QPSO), the search space and solution space of problem are different qualities. Wave function or probability function of position depicts the state of the particle in quantized search space [41]. The fifth is CLPSO, which uses other particles' historical best information to update the target particle's velocity. It is a simple and well-performed algorithm for solving the multimodal problems [22]. The sixth is the fully informed PSO (FIPS) with ring topology [27]. The seventh is a dynamic multi-swarm PSO (DMSPSO), which adopts a dynamic way to regroup the swarm and changes the topology structure periodically [23]. The eighth is a fitness-distance-ratio-based PSO (FDRPSO), which is accomplished by using the ratio of the relative fitness and the distance of other particles to determine the direction in which each component of the particle position needs to be changed [32]. These typical PSO variants are chosen for comparisons with the proposed VCPSO algorithm. For a fair comparison among all mentioned PSOs, they were tested using the same population size and maximum fitness evaluations (FEs) in each run. The total number of function evaluations equals D^*10^4 (D is the searching dimension). Therefore, when solving the 10-D problem, the population size is 40 and the number of iterations in each run 2.5×10^3. When solving the 30-D problem, the population size is 40 and the number of iterations in each run is 7.5×10^3, et.al. All benchmark functions were run 30 times independently aiming to reduce statistical errors, and the average testing results were used for comparison. To



(a) F_1 (b) F_4 (c) F_7 (d) F_8

(e) F_9 (f) F_{12} (g) F_{14} (h) F_{15}

Fig. 5. Comparison the effects of different minimum interval size m for convergence accuracy on 10-D dimensional problems.



(a) F_1 (b) F_3 (c) F_4 (d) F_8

(e) F_9 (f) F_{13} (g) F_{14} (h) F_{15}

Fig. 6. Comparison the effects of different minimum interval size m for convergence accuracy on 30-D dimensional problems.

verify whether the results generated by VCPSO are significantly different from the compared algorithms, we use the non-parametric statistical Wilcoxon rank sum test [13] to perform rigorous comparisons between VCPSO and its peers. The test was conducted at 5% significance level. The value of h indicates whether the performance of VCPSO is better (i.e., $h =$ "+"), insignificant(i.e., $h =$ "="), or worse ((i.e., $h =$ "-")) than the compared algorithm at the statistical level.

4.3.2. Numerical results on unimodal functions

In this subsection, we test the performance of VCPSO on unimodal functions. These functions contain the basic unimodal functions (F_1–F_6) and the shifted version (F_{16}–F_{19}). All of these functions were tested on 10-D and 30-D dimensional respectively. Tables 5 and 6 show the 10-D and 30-D tests result in terms of the mean best fitness *Mean*, the standard deviations *StD*. and the significance level indicator *h*. The lowest values in each line for the *mean* value are highlighted in boldface. Moreover, the convergence curves of VCPSO and other algorithm are shown for 30-D problems Fig. 7. The 10-D convergence curves are similar to corresponding 30-D problems, thus they are not shown repeatedly.

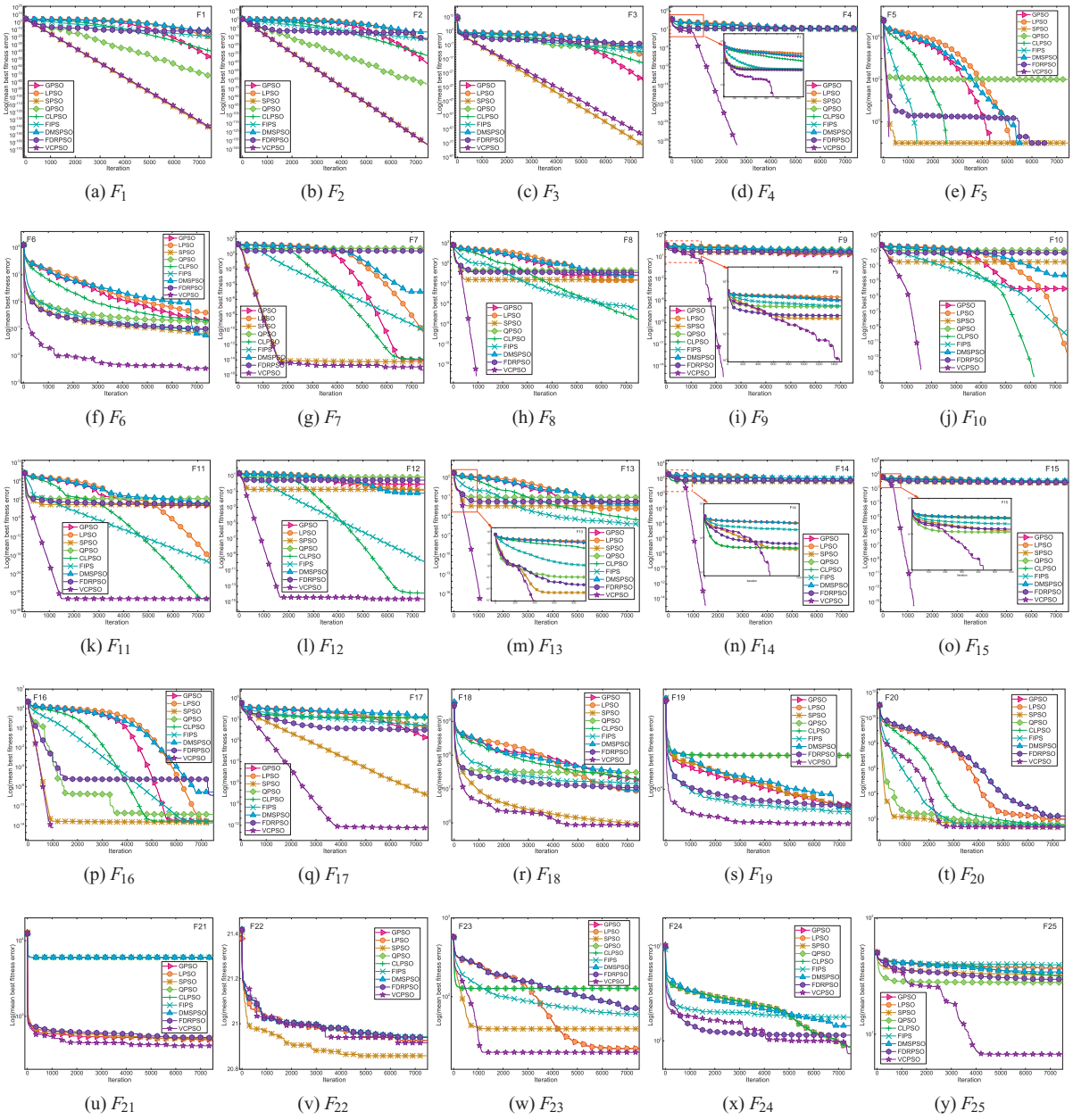
From the statistical results and convergence curves, one can see that the VCPSO algorithm yields better results on mean beset fitness than other compared algorithms for a majority of test function such as F_4, F_5, F_6, F_{16}–F_{19}. The algorithm

Table 5
Mean, StD., and *h* value for 10-D problem on unimodal functions.

Func.	Item	GPSO	LPSO	SPSO	QPSO	CLPSO	FIPS	DMSPSO	FDRPSO	**VCPSO**
F_1	*Mean*	4.25E−64	8.46E−33	4.72E−115	**7.21E−129**	4.43E−39	1.37E−24	1.65E−24	3.22E−68	3.74E−85
	StD.	1.90E−63	4.51E−31	1.39E−114	3.22E−128	3.85E−39	4.36E−24	3.28E−24	5.31E−68	1.43E−84
	h	+	+	−	−	+	+	+	+	
F_2	*Mean*	1.64E−63	7.18E−32	2.79E−112	**3.89E−122**	7.90E−38	1.11E−23	7.13E−23	1.20E−66	3.05E−84
	StD.	7.19E−63	1.87E−31	2.81E−113	1.74E−121	2.08E−37	4.95E−24	2.47E−22	5.36E−66	9.04E−84
	h	+	+	−	−	+	+	+	+	
F_3	*Mean*	8.81E−38	6.81E−20	**5.59E−59**	4.47E−29	1.09E−22	5.20E−14	7.48E−15	3.07E−24	6.66E−44
	StD.	6.47E−37	1.78E−20	1.78E−58	2.00E−28	1.73E−22	9.12E−14	1.57E−14	1.37E−23	4.48E−43
	h	+	+	−	+	+	+	+	+	
F_4	*Mean*	2.85E+00	3.72E+00	1.20E−02	1.49E+00	4.29E+00	3.35E+00	4.77E+00	3.21E+00	**0.00E+00**
	StD.	4.36E−03	9.47E+00	2.24E−02	6.64E+00	2.10E+00	1.70E−01	8.76E−01	3.02E+00	**0.00E+00**
	h	+	+	+	+	+	+	+	+	
F_5	*Mean*	**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**
	StD.	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
	h	=	=	=	=	=	=	=	=	
F_6	*Mean*	8.58E−03	1.57E−02	7.23E−03	9.48E−03	1.41E−02	7.46E−03	4.54E−03	5.23E−03	**5.62E−04**
	StD.	2.69E−02	2.58E−02	5.47E−03	7.85E−03	2.11E−02	9.74E−03	1.04E−02	1.39E−02	1.03E−03
	h	+	+	+	+	+	+	+	+	
F_{16}	*Mean*	3.41E−14	1.36E−07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	StD.	7.19E−14	3.52E−07	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	h	+	+	=	=	=	=	=	=	
F_{17}	*Mean*	3.41E−14	2.13E−07	**0.00E+00**	9.57E−10	5.71E−03	2.89E−08	6.50E−02	6.25E−14	**0.00E+00**
	StD.	1.08E−13	5.64E−07	0.00E+00	3.03E−09	1.32E−03	5.25E−09	1.68E−01	1.80E−14	0.00E+00
	h	+	+	=	+	+	+	+	+	
F_{18}	*Mean*	2.79E+05	1.11E+05	1.57E+05	1.81E+05	5.65E+05	1.10E+05	4.94E+04	4.73E+05	**1.45E+04**
	StD.	7.57E+05	1.57E+05	6.20E+04	3.95E+05	3.58E+05	1.19E+05	1.76E+04	1.01E+06	8.99E+03
	h	+	+	+	+	+	+	+	+	
F_{19}	*Mean*	1.16E+03	1.11E+03	**0.00E+00**	3.80E+03	1.27E+03	1.15E+03	9.70E+02	1.45E+03	8.30E+02
	StD.	6.20E+02	1.54E+02	0.00E+00	3.90E+02	4.56E+02	5.97E+02	2.04E+02	2.57E+02	2.64E+02
	h	+	+	−	+	+	+	+	+	
	w/t/l	9/1/0	9/1/0	3/3/4	6/1/3	8/2/0	8/2/0	8/2/0	8/2/0	−/−/−

Table 6
Mean, StD., and *h* value for 30-D problem on unimodal functions.

Func.	Item	GPSO	LPSO	SPSO	QPSO	CLPSO	FIPS	DMSPSO	FDRPSO	**VCPSO**
F_1	*Mean*	3.43E−48	1.81E−20	**1.52E−142**	1.44E−71	5.62E−39	1.55E−19	1.17E−10	1.96E−15	5.61E−141
	StD.	9.64E−48	8.93E−20	2.25E−141	4.54E−71	1.35E−38	3.78E−20	3.65E−10	5.46E−14	5.86E−141
	h	+	+	−	+	+	+	+	+	
F_2	*Mean*	7.42E−47	3.04E−19	6.59E−140	1.20E−69	4.15E−37	3.78E−18	1.87E−09	7.39E−17	**2.39E−140**
	StD.	1.39E−45	6.99E−19	2.08E−139	3.79E−69	3.41E−37	1.03E−18	1.71E−09	2.53E−17	3.51E−140
	h	+	+	+	+	+	+	+	+	
F_3	*Mean*	1.40E−32	9.10E−15	**1.51E−78**	3.16E−08	2.95E−20	3.47E−12	1.35E−08	1.51E−06	1.33E−71
	StD.	3.81E−32	1.10E−14	1.95E−78	9.99E−08	9.31E−20	5.08E−13	6.48E−08	4.54E−06	2.16E−71
	h	+	+	−	+	+	+	+	+	
F_4	*Mean*	2.27E+01	2.23E+01	1.13E+01	1.46E+01	2.21E+01	2.34E+01	2.49E+01	2.28E+01	**0.00E+00**
	StD.	2.50E+01	4.44E+00	3.65E+00	2.71E+00	6.84E+00	3.83E−02	2.65E+00	3.55E+00	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_5	*Mean*	**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**
	StD.	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
	h	=	=	=	=	=	=	=	=	
F_6	*Mean*	1.97E−01	3.80E−01	6.58E−02	1.80E−01	2.03E−01	9.35E−02	5.54E−02	1.01E−01	**3.42E−03**
	StD.	7.42E−02	5.72E−01	1.17E−01	5.54E−02	2.88E−01	9.23E−02	9.03E−02	6.63E−01	2.12E−03
	h	+	+	+	+	+	+	+	+	
F_{16}	*Mean*	5.68E−14	5.68E−14	2.84E−14	6.20E−13	5.68E−14	3.69E−14	1.64E−09	4.52E−10	**0.00E+00**
	StD.	0.00E+00	0.00E+00	1.27E−13	1.80E−12	5.68E−14	8.90E−14	6.65E−09	2.02E−09	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{17}	*Mean*	2.02E+00	1.63E−08	1.73E+03	8.10E+02	1.36E+02	2.69E−07	2.54E+03	3.13E+01	**4.26E−13**
	StD.	1.94E+00	7.73E−08	1.25E+04	9.26E+00	5.92E+01	5.00E−07	6.08E+02	7.36E+01	8.99E−14
	h	+	+	+	+	+	+	+	+	
F_{18}	*Mean*	1.83E+07	8.50E+06	9.11E+05	3.10E+07	1.95E+07	1.46E+07	9.31E+06	1.09E+07	**8.63E+05**
	StD.	1.28E+07	8.86E+06	8.57E+05	2.14E+07	2.95E+05	4.65E+06	5.94E+05	7.57E+06	3.25E+05
	h	+	+	+	+	+	+	+	+	
F_{19}	*Mean*	7.55E+03	7.07E+03	**2.11E+03**	1.83E+04	7.69E+03	6.68E+03	7.22E+03	7.50E+03	5.41E+03
	StD.	2.23E+03	2.11E+03	7.24E+02	8.04E+03	2.81E+03	2.04E+03	3.31E+01	2.34E+03	5.97E+02
	h	+	+	−	+	+	+	+	+	
	w/t/l	9/1/0	9/1/0	6/1/3	9/1/0	9/1/0	9/1/0	9/1/0	9/1/0	−/−/−



(a) F_1 (b) F_2 (c) F_3 (d) F_4 (e) F_5

(f) F_6 (g) F_7 (h) F_8 (i) F_9 (j) F_{10}

(k) F_{11} (l) F_{12} (m) F_{13} (n) F_{14} (o) F_{15}

(p) F_{16} (q) F_{17} (r) F_{18} (s) F_{19} (t) F_{20}

(u) F_{21} (v) F_{22} (w) F_{23} (x) F_{24} (y) F_{25}

Fig. 7. The convergence curves of benchmark functions on 30-D dimensional problems.

with global learning strategy showed a fast convergence on F_1 and F_2 such as GPSO, SPSO, FDRPSO However, from the convergence curves and the mean best accuracy, VCPSO still can be comparable with them as it introduced the global version operator, namely, the centralized operator, to accelerate the swarm convergence. For complex unimodal problem especially the shifted version, VCPSO algorithm showed the best performance. It surpasses other algorithms on complex unimodal problems from F_4–F_6, and F_{16}–F_{19}. Specially, among these unimodal functions, the Rosenbrock's function F_4 can also be treated as a multimodal problems, because it has a narrow valley between the perceived local optima and the global optimum, it's very difficult to get the optimum position for most of the state-of-art optimization algorithms. However, VCPSO can find the global optimum robustly and efficiently owning to the convolving operators that enrich the population diversity and help particles escape from local optima.

In addition, the statistical results is provided by using a two-sample Wilcoxon rank sum test. The nonparametric statistical test can distinguish the statistical significance of difference between two independent individuals. As can be seen in Tables 5 and 6, the value of h calculated on most functions showed that the results obtained by VCPSO and the ones acquired by other algorithms are statistical significant.

Table 7
Mean, StD., and *h* value for 10-D problem on the multimodal functions.

Func.	Item	GPSO	LPSO	SPSO	QPSO	CLPSO	FIPS	DMSPSO	FDRPSO	**VCPSO**
F_7	*Mean*	4.00E−15	4.00E−15	3.64E−15	7.86E−01	4.00E−15	4.58E−13	6.66E−13	4.00E−15	**4.44E−16**
	StD.	0.00E+00	0.00E+00	1.59E−15	1.65E+00	0.00E+00	1.49E−12	2.36E−12	0.00E+00	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_8	*Mean*	6.95E−02	4.28E−02	2.29E−02	1.66E−01	6.23E−02	3.14E−02	1.43E−01	4.87E−02	**0.00E+00**
	StD.	4.11E−02	2.87E−02	3.64E−02	2.70E−01	2.56E−01	3.89E−02	2.46E−01	2.25E−03	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_9	*Mean*	1.14E+00	2.70E+00	2.89E+00	2.08E+01	2.60E+00	**3.16E−01**	7.20E+00	3.20E+00	1.04E+00
	StD.	6.67E−01	1.28E+00	4.00E+00	1.84E+01	2.90E+00	3.16E+00	5.04E+00	3.48E+00	9.35E+00
	h	+	+	+	+	+	−	+	+	
F_{10}	*Mean*	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.27E−01	0.00E+00	8.53E−15	7.13E−12	3.00E−03	**0.00E+00**
	StD.	0.00E+00	0.00E+00	0.00E+00	1.68E+00	0.00E+00	6.36E−15	2.48E−11	1.34E−02	0.00E+00
	h	=	=	=	+	+	+	+	+	
F_{11}	*Mean*	4.71E−32	5.20E−32	**4.71E−32**	3.41E−01	4.71E−32	5.06E−26	3.50E−24	4.71E−32	**4.71E−32**
	StD.	2.45E−47	2.16E−32	2.45E−47	1.52E+00	2.45E−47	1.89E−25	1.63E−22	2.45E−47	1.22E−47
	h	+	+	=	+	+	+	+	+	
F_{12}	*Mean*	4.00E−15	4.00E−15	3.64E−15	1.45E+00	4.00E−15	4.60E−13	1.49E−12	1.16E−01	**4.44E−16**
	StD.	0.00E+00	0.00E+00	1.43E−14	6.49E+00	0.00E+00	6.36E−15	6.34E−12	5.17E−01	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{13}	*Mean*	1.49E−01	4.63E−02	3.38E−02	1.43E−01	1.51E−01	6.15E−02	1.44E−01	9.20E−02	**0.00E+00**
	StD.	2.27E−01	2.41E−02	6.29E−02	1.42E−01	7.98E−02	1.38E−01	9.99E−02	7.37E−02	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{14}	*Mean*	8.76E+00	6.57E+00	6.25E+00	1.05E+01	1.08E+01	7.17E+00	1.03E+01	7.52E+00	**0.00E+00**
	StD.	1.87E+01	6.23E+00	1.21E+01	1.16E+01	9.91E+00	9.81E+00	1.09E+00	2.43E+00	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{15}	*Mean*	1.27E+01	9.38E+00	7.18E+00	1.77E+01	1.04E+01	6.84E+00	1.40E+01	9.14E+00	**0.00E+00**
	StD.	7.71E−01	1.06E+01	2.20E+01	3.44E+01	5.32E+00	9.06E−01	1.79E+01	1.61E+01	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{20}	*Mean*	1.75E+01	3.35E+00	8.34E−01	2.37E+01	7.17E+00	4.03E+00	5.47E+00	1.39E+01	**7.61E−06**
	StD.	5.01E+01	9.88E+00	2.60E+00	6.22E+01	5.83E+00	1.74E+00	3.81E+01	3.32E+01	1.64E−05
	h	+	+	+	+	+	+	+	+	
F_{21}	*Mean*	5.88E+01	5.43E+01	1.27E+03	1.37E+03	4.94E+01	4.88E+01	5.25E+01	5.41E+01	**3.89E+01**
	StD.	1.90E+01	2.64E−01	3.99E−01	1.36E+02	5.18E+00	6.04E+01	4.64E+01	3.41E+00	1.42E+00
	h	+	+	+	+	+	+	+	+	
F_{22}	*Mean*	2.04E+01	**2.03E+01**	**2.03E+01**	2.04E+01	**2.03E+01**	2.04E+01	**2.03E+01**	2.03E+01	**2.03E+01**
	StD.	3.42E−01	2.59E−02	1.39E−01	1.41E−01	2.31E−01	2.91E−01	2.11E−02	5.17E−01	3.06E−01
	h	+	−	−	+	−	+	−	−	
F_{23}	*Mean*	2.19E+00	2.69E+00	3.68E+00	1.46E+01	1.77E+00	**8.73E−01**	6.73E+00	3.99E+00	1.69E+00
	StD.	6.29E−01	9.39E−01	2.20E+00	2.11E+01	4.74E−02	6.82E+00	7.86E+00	9.45E+00	9.44E−01
	h	+	+	+	+	+	−	+	+	
F_{24}	*Mean*	1.01E+01	7.70E+00	7.10E+00	2.38E+01	1.48E+01	1.28E+01	1.34E+01	1.54E+01	**2.39e+000**
	StD.	3.78E+00	8.60E+00	4.01E+00	3.46E+00	6.22E−01	2.10E+01	3.90E+00	1.72E+01	8.89e−001
	h	+	+	+	+	+	+	+	+	
F_{25}	*Mean*	4.89E+00	4.54E+00	5.56E+00	5.55E+00	6.83E+00	5.62E+00	6.27E+00	5.10E+00	**1.20E+00**
	StD.	1.76E−01	5.83E+00	9.71E−01	6.18E+00	2.75E+00	1.01E+01	1.26E+00	9.05E−02	4.02E−01
	h	+	+	+	+	+	+	+	+	
F_{26}	*Mean*	1.15E+03	5.91E+00	2.50E+02	1.81E+02	5.78E+02	4.60E+01	6.30E+01	1.22E+03	**4.03E+00**
	StD.	3.64E+03	1.88E+01	7.57E+02	5.39E+02	6.31E+01	5.89E+01	3.50E+01	3.79E+03	8.70E+00
	h	+	+	+	+	+	+	+	+	
F_{27}	*Mean*	7.15e−001	**6.26e−001**	8.39E−01	8.99E−01	1.16E+00	1.27E+00	1.16E+00	6.65E−01	9.45e−001
	StD.	1.22e+00	1.91e−001	2.85E−01	2.78E−01	5.38E−02	1.97E−01	2.50E−01	1.05E−01	6.42e−001
	h	−	−	−	−	+	+	+	−	
F_{28}	*Mean*	2.55E+00	2.67E+00	2.82E+00	2.64E+00	3.36E+00	3.00E+00	3.01E+00	2.74E+00	**2.00E+00**
	StD.	4.33E−01	3.76E−01	2.04E+00	1.14E+00	5.70E−01	1.26E+00	8.20E−01	5.14E−01	1.15E+00
	h	+	+	+	+	+	+	+	+	
	w/t/l/	16/1/1	15/1/2	14/2/2	17/0/1	17/0/1	16/0/2	17/0/1	16/0/2	−/−/−

4.3.3. Numerical results on basic and complex multimodal functions

For the multimodal functions, there are many local optima in the search space but can either have a single or more than one global optima. The search algorithms may be easily fall into local optima especially when the global optimum is shifted or rotated. The test multimodal functions in Table 1 contains the basic multimodal function (F_7–F_{11}), complex rotated functions (F_{12}–F_{15}) and shifted functions (F_{20}–F_{28}). The experiments were conducted on these multi-modal functions on the 10-D and 30-D separately. Tables 7 and 8 present the results in terms of the fitness means (*Mean*), standard deviations (*StD.*), and the significant indicator *h* on 10 and 30 dimensional. Moreover, the convergence curves of VCPSO and other algorithms on 30-D problems are shown in Fig. 7. From the results in both tables, the proposed VCPSO algorithm produce the competitive results for most of the multimodal functions. As the VCPSO algorithm optimize the sub-dimensions with co-evolving operators rather than optimization the full dimension with a single strategy, so it has a low probability to fall into the optima.

Table 8
Mean, StD., and *h* value for 30-D problem on the multimodal functions.

Func.	Item	GPSO	LPSO	SPSO	QPSO	CLPSO	FIPS	DMSPSO	FDRPSO	**VCPSO**
F_7	*Mean*	9.86E−15	3.83E−11	5.95E−15	5.92E+00	1.32E−14	9.52E−11	1.07E−05	2.71E+00	**4.44E−16**
	StD.	1.03E−14	5.54E−10	7.15E−15	8.10E+00	6.36E−15	8.33E−12	3.53E−05	2.84E+00	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_8	*Mean*	2.09E−02	3.45E−03	4.44E−03	1.02E−01	5.45E−09	1.54E−07	1.75E−02	5.17E−02	**0.00E+00**
	StD.	1.37E−01	3.97E−02	1.98E−02	4.08E−01	2.20E−08	6.90E−07	3.40E−02	2.31E−01	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_9	*Mean*	2.25E+01	3.06E+01	3.85E+01	1.10E+02	2.76E+01	6.06E+01	7.42E+01	4.63E+01	**0.00E+00**
	StD.	7.34E+00	3.17E+00	3.27E+01	1.54E+01	7.98E+00	4.26E+01	4.75E+01	3.78E+01	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{10}	*Mean*	1.03E−04	2.68E−13	3.25E−01	1.10E+01	**0.00E+00**	6.84E−11	5.94E−03	5.02E+00	**0.00E+00**
	StD.	3.15E−04	1.20E−12	6.03E−01	5.18E+00	0.00E+00	1.14E−10	2.98E−02	9.96E−01	0.00E+00
	h	+	+	+	+	=	+	+	+	
F_{11}	*Mean*	2.07E−02	1.63E−19	3.63E−02	2.86E+00	1.57E−32	7.87E−21	1.03E−01	7.26E−02	**1.57E−32**
	StD.	9.27E−02	6.96E−19	1.62E−01	1.09E+01	1.22E−47	1.39E−20	4.60E−01	6.03E−01	0.00E+00
	h	+	+	+	+	=	+	+	+	
F_{12}	*Mean*	8.73E−01	1.47E−01	1.87E−01	6.62E+00	1.36E−14	1.22E−10	6.35E−02	2.73E+00	**2.58E−15**
	StD.	3.91E+00	6.56E−01	8.35E−01	3.46E+00	4.77E−15	3.87E−11	2.84E−01	7.56E+00	3.18E−15
	h	+	+	+	+	+	+	+	+	
F_{13}	*Mean*	2.15E−02	1.60E−03	4.56E−03	9.64E−02	2.93E−05	7.23E−06	9.77E−03	2.11E−02	**0.00E+00**
	StD.	9.02E−02	7.16E−03	2.04E−02	1.47E−01	6.81E−05	3.23E−05	1.05E−02	6.11E−02	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{14}	*Mean*	6.19E+01	4.54E+01	4.44E+01	5.04E+01	6.77E+01	1.15E+02	9.31E+01	6.15E+01	**0.00E+00**
	StD.	2.76E+01	6.22E+00	1.78E+00	1.33E+00	2.09E+01	5.39E+01	1.05E+02	7.46E+01	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{15}	*Mean*	7.48E+01	6.96E+01	7.16E+01	1.14E+02	7.37E+01	1.14E+02	1.21E+02	8.00E+01	**0.00E+00**
	StD.	9.37E+01	2.47E+01	1.98E+01	1.42E+00	1.26E+01	2.94E+01	4.23E+01	2.51E+01	0.00E+00
	h	+	+	+	+	+	+	+	+	
F_{20}	*Mean*	9.79E+01	4.40E+01	**2.12E+01**	3.92E+01	2.83E+01	3.19E+01	1.76E+02	9.79E+01	2.35E+01
	StD.	2.95E+02	6.04E+01	6.63E+01	8.64E+01	1.65E+01	2.07E+01	1.69E+02	1.76E+02	7.71E−03
	h	+	+	−	+	+	+	+	+	
F_{21}	*Mean*	4.92E+02	5.08E+02	4.71E+03	6.01E+03	4.99E+02	4.98E+02	5.21E+02	5.19E+02	**4.08E+02**
	StD.	1.72E+02	1.11E+02	4.61E+01	3.87E+02	1.26E+02	1.73E+02	8.90E+01	2.67E+02	3.75E+00
	h	+	+	+	+	+	+	+	+	
F_{22}	*Mean*	2.09E+01	2.10E+01	2.09E+01	2.09E+01	2.09E+01	2.10E+01	2.10E+01	**2.08E+01**	2.09E+01
	StD.	1.99E−01	8.89E−02	1.90E−01	2.48E−01	1.31E−01	2.99E−01	1.58E−02	4.09E−02	4.56E−02
	h	+	+	−	+	+	+	+	−	
F_{23}	*Mean*	2.20E+01	2.83E+01	3.91E+01	1.28E+02	2.35E+01	5.93E+01	7.13E+01	7.35E+01	**1.97E+01**
	StD.	2.83E+00	2.08E+01	7.24E+00	1.64E+02	9.50E−01	1.99E+01	1.63E+01	2.73E−01	4.45E−01
	h	+	+	+	+	+	+	+	+	
F_{24}	*Mean*	8.58E+01	1.06E+02	**5.10E+01**	2.49E+02	1.33E+02	1.78E+02	1.45E+02	1.15E+02	7.30E+01
	StD.	5.11E+01	1.47E+02	2.59E+01	3.12E+01	1.66E+01	4.46E+01	1.17E+01	1.83E+02	1.21E+02
	h	+	+	−	+	+	+	+	+	
F_{25}	*Mean*	3.70E+01	3.84E+01	3.21E+01	2.79E+01	3.15E+01	3.95E+01	3.43E+01	2.96E+01	**6.74E+00**
	StD.	3.38E+00	7.65E+00	5.36E+00	2.38E+00	2.07E+00	2.11E+00	1.07E+00	9.27E+00	4.49E+00
	h	+	+	+	+	+	+	+	+	
F_{26}	*Mean*	3.73E+04	2.12E+04	**1.02E+04**	2.89E+04	4.74E+04	4.54E+04	1.05E+05	4.52E+04	1.56E+04
	StD.	7.62E+04	1.66E+04	2.34E+04	1.97E+05	3.35E+04	6.43E+04	5.52E+04	5.63E+04	2.02E+03
	h	+	+	−	+	+	+	+	+	
F_{27}	*Mean*	3.84E+00	4.06E+00	4.20E+00	2.08E+01	8.02E+00	1.24E+01	1.02E+01	5.21E+00	**2.55E+00**
	StD.	1.97E+00	3.40E+00	3.97E+00	2.70E+01	2.96E+00	4.74E−01	1.78E+00	4.46E−01	1.12E+00
	h	+	+	+	+	+	+	+	+	
F_{28}	*Mean*	1.32E+01	1.30E+01	1.30E+01	1.23E+01	1.31E+01	1.31E+01	1.30E+01	1.24E+01	**1.16E+01**
	StD.	6.26E−01	2.31E+00	9.10E−01	1.39E+00	5.27E−01	6.64E−01	3.73E−01	2.06E+00	9.16E−01
	h	+	+	+	+	+	+	+	+	
	w/t/l/	18/0/0	18/0/0	14/0/4	18/0/0	16/2/0	18/0/0	18/0/0	17/0/1	−/−/−

When solving higher dimension problems, the search for the global optimum position generally became more difficult than the lower dimension. In such case, a more fitness evaluations may be produce a better results. Thus, in our tests, when solving the 30-D problems, the number of fitness evaluations is setting to 30×10^4 rather than the 10-D problems settings by 10×10^4. The results obtained by the compared algorithms were not as good as in the 10-D counterpart. Conversely, VCPSO can still reach or approximate to the global optimum and specially for the unimodal problems such as F_1, F_2, F_3, and F_4. The results indirectly reflect that VCPSO have the dimensional insensitivity characteristics.For multimodal functions from F_7 to F_{12}, VCPSO generally outperforms all of the other PSO variants on function F_7, F_8, F_9, F_{11}, and F_{12}, and significantly improves the results on function F_8 and F_9. Among these PSO variants, CLPSO also showed good searching abilities on multimodal problems. F_{10} is a noncontinuous function constructed based on the Rastrigin's function, and it has many local optimum points, it is easily trapped into the optimum position for all algorithms. However, VCPSO can still

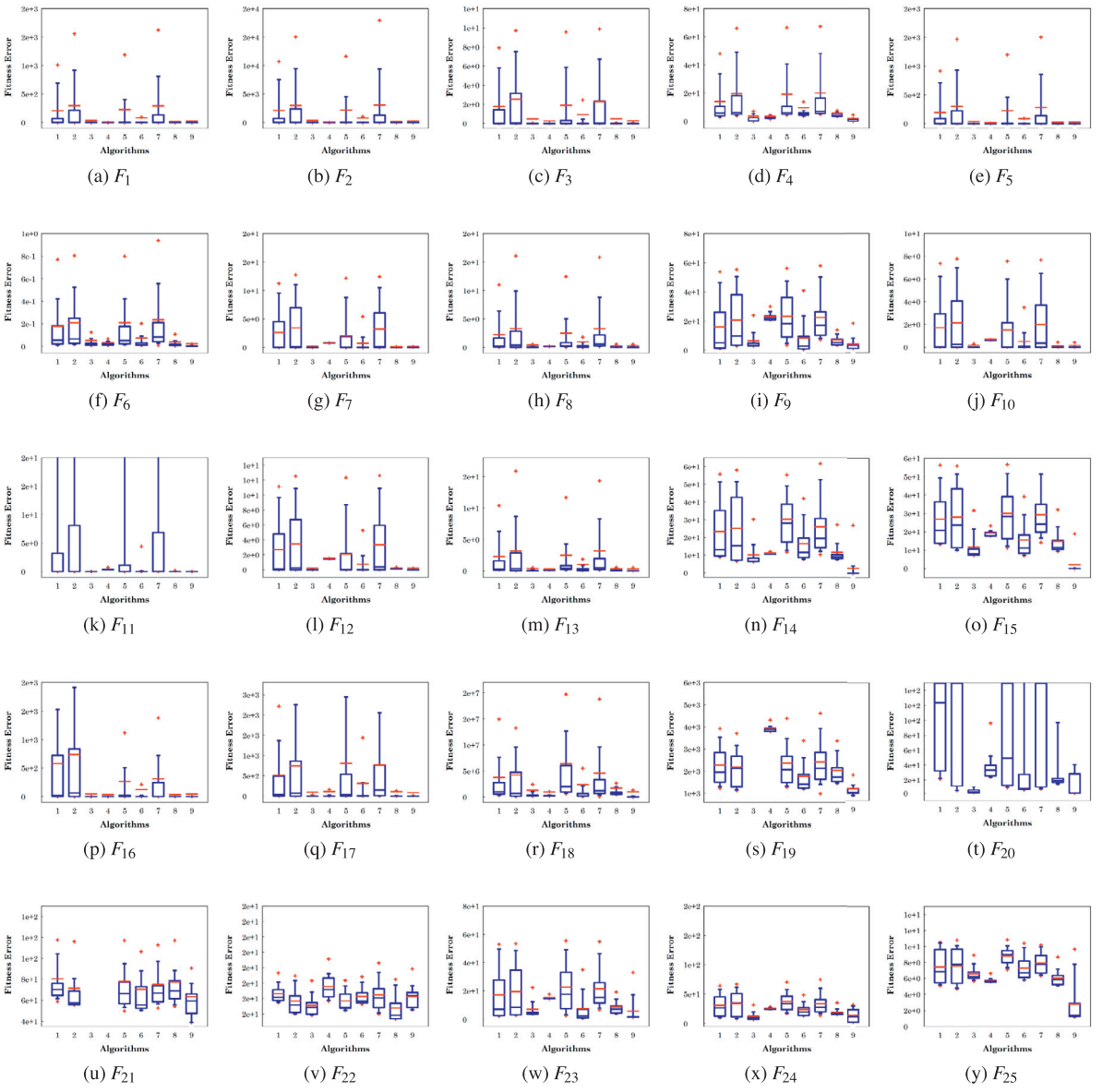


Fig. 8. The box plot of benchmark functions.

get the optimum position with a high probability during the repeat tests. In addition, the complex Schwefel's function F_{11} traps most algorithms in local optima, VCPSO together with CLPSO and DMS-PSO successfully avoid falling into the deep local optimum which is far from the global optimum. For rotated multimodal functions from F_{20} to F_{28}, VCPSO can find the optima of all of the rotated multimodal functions in the test and showed significantly better performance than the other PSO algorithms. The convergence processes of these compared algorithms are listed in Fig. 7. Comparing the results and the convergence curves of these ten algorithms, SPSO generally converges faster than GPSO and LPSO, FIPS are all local version of PSO, they generally performance better than the global versions. FIPS yields a comparatively better performance than LPSO. DMSPSO performs comparable with CLPSO on unimodal and multimodal problems, FDRPSO has a good local search ability. However, all of them were seriously affected when the test functions are rotated. Among these algorithms, the test results showed that VCPSO was still the best algorithm on these complex rotated function. It converged slower at the begin of the process, but faster in the late process and achieved a higher accuracy than the other algorithms especially in functions 4, 9, 14, 15, 16. The box plot of 10 dimensional problems on functions from F_1 to F_{25} are listed in Fig. 8. From the box plot, one can see that VCPSO performs better on most functions than the compared algorithms. Overall, VCPSO surpassed or was comparable to all of the other algorithms on functions F_7–F_{22}, F_{23} and F_{27}. The test results and search performance confirmedly demonstrate the effectiveness and scalability of VCPSO in solving different searching problems.

Table 9
Summary of the CEC2015 learning-based benchmark functions.

Function categories	No.	Functions	Search range	$F_i^* = F(x^*)$
Unimodal Functions	TF_1	Rotated High Conditioned Elliptic Function	[−100,100]	100
	TF_2	Rotated Cigar Function	[−100,100]	200
Multimodal Ffunctions	TF_3	Shifted and Rotated Ackleys Function	[−100,100]	300
	TF_4	Shifted and Rotated Rastrigins Function	[−100,100]	400
	TF_5	Shifted and Rotated Schwefels Function	[−100,100]	500
Hybrid Ffunctions	TF_6	Hybrid Function 1 ($N = 3$)	[−100,100]	600
	TF_7	Hybrid Function 2 ($N = 4$)	[−100,100]	700
	TF_8	Hybrid Function 3 ($N = 5$)	[−100,100]	800
Composition Ffunctions	TF_9	Composition Function 1 ($N = 3$)	[−100,100]	900
	TF_{10}	Composition Function 2 ($N = 3$)	[−100,100]	1000
	TF_{11}	Composition Function 3 ($N = 5$)	[−100,100]	1100
	TF_{12}	Composition Function 4 ($N = 5$)	[−100,100]	1200
	TF_{13}	Composition Function 5 ($N = 5$)	[−100,100]	1300
	TF_{14}	Composition Function 6 ($N = 7$)	[−100,100]	1400
	TF_{15}	Composition Function 7 ($N = 10$)	[−100,100]	1500

4.3.4. Numerical results on non-separable problems

In this subsection, the performance of the compared algorithms are tested by solving a set of non-separable problems taken from the CEC2015 competition on learning based real-parameter single objective optimization [24]. The main difference between the CEC2015 functions and the previous CEC2005/2013/2014 is that the CEC2015 test suite consists of learning-based functions which have several shift vectors and rotation matrices. Thus, it's very hard to get the optimum solutions with minor error. The names and attributes of CEC2015 non-separable functions ($TF_1 \sim TF_{15}$) are given in Table 9. The test set contains 15 problems. These problems have different mathematical properties on different testing dimensions. Functions $TF_1 \sim TF_2$ are shifted unimodal functions, $TF_3 \sim TF_5$ are shifted and rotated multimedia functions, $TF_6 \sim TF_8$ are hybrid functions, and $TF_9 \sim TF_{15}$ are composite functions which combine multiple test problems into a complex landscape. All of these CEC2015 functions are non-separable functions. Generally, In the area of numerical optimization, the optimized functions can be classified into two categories: separable functions and non-separable functions. For example, if the decision variables involved in a problem are independent of each other, the problem can be easily solved by decomposing it into several sub-problems, each of which involving only one decision variable while treating all others as constants. This way, even a line search or greedy method can solve the problem efficiently. This class of problem $f(x)$is known as a separable problem, and can be defined as follows in Eq. (11):

$$\underset{(x_1,\ldots,x_n)}{arg\ min} = (\underset{x_1}{arg\ min} f(x_1,\cdots),\cdots,\underset{x_n}{arg\ min} f(\cdots,x_n)) \tag{11}$$

In other words, a function of n variables is separable if it can be rewritten as a sum of n functions of just one variable for each function. If a function $f(x)$ is separable, its parameters x_i are called independent. Functions which are not separable are called non-separable.

In the test, the centralized operator and decentralized operator were selected as the update rule to optimize the non-separable problems. The test experiments were conducted on CEC'2015 benchmark functions for 10 dimensional and 30 dimensional problems. For each D-dimensional problem instance, the search space is $[-100, 100]^D$. Each function was evaluated 10^4*D times. All of the parameter settings for the compared algorithms are based on the suggestions in Table 4. The absolute error value of the found solution, $\mathbf{x}$, is defined as: $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $\mathbf{x}^*$ is the optimum value of the target function f.

The experiments were conducted on these functions on the 10-D and 30-D respectively. Tables 10 and 11 present the results of twenty-five runs of the compared algorithms in terms of the fitness means (*Mean*), standard deviations (*StD.*), and the significant indicator h on 10 and 30 dimensional. The top two mean fitness values are shown in bold. From the tests results, the proposed algorithm produce the significant competitive performance on most functions, such as TF_1, TF_2, TF_4, TF_5, TF_6, TF_7, TF_9, TF_{10}, TF_{13}, TF_{15} for 10-D problems and TF_1, TF_2, TF_3, TF_4, TF_5, TF_6, TF_8, TF_9, TF_{12}, TF_{14}, TF_{15} for 30-D problems. To get a statistical conclusion of the experimental results, the two-tailed Wilcoxon rank sum tests with significance level = 0.05 is conducted to judge the significance of the performance between VCPSO and its competitor. The cases are labeled with " +"," =", and " −" when the performance of VCPSO is significantly better than, similar to and worse than the compared algorithm.

The pairwise comparison results between VCPSO and its peers by using Wilcoxon signed ranks sum test are summarized in Tables 10 and 11. These tables provide the comparison results for each test functions by using the h values. From the summary results, it can be seen that the number of problems that VCPSO outperforms its peers is much larger than the number of problems in which it significantly performs worse than its peers. The whole summary results of significance results in the last row in Tables 10 and 11 confirm the significant improvements of VCPSO over the selected compared PSO variants.

Table 10
Mean, and standard deviation for 10-dimensional of CEC2015 benchmark functions TF_1–TF_{15}.

Func.	item	GPSO	LPSO	SPSO	QPSO	CLPSO	FIPS	DMSPSO	FDRPSO	VCPSO
TF_1	*Mean*	5.046E+06	1.173E+05	2.779E+04	3.445E+05	5.527E+05	2.512E+05	9.637E+04	**3.790E+04**	**3.110E+03**
	StD.	2.307E+07	7.303E+04	1.014E+04	1.747E+05	7.419E+04	3.499E+04	2.458E+05	5.299E+04	6.937E+02
	h	+	+	+	+	+	+	+	+	
TF_2	*Mean*	6.015E+08	2.944E+07	6.148E+03	6.496E+03	4.727E+04	6.487E+03	1.376E+04	7.357E+03	**4.517E+03**
	StD.	1.046E+09	7.788E+07	2.826E+04	7.161E+03	8.119E+04	3.856E+02	5.467E+03	2.293E+04	2.526E+04
	h	+	+	+	+	+	+	+	+	
TF_3	*Mean*	2.027E+01	2.019E+01	2.019E+01	2.032E+01	2.023E+01	2.034E+01	**2.010E+01**	**2.018E+01**	2.033E+01
	StD.	1.115E−01	4.072E−02	2.993E−02	6.276E−03	2.413E+00	1.506E−01	1.640E−01	1.427E−01	2.294E−01
	h	−	−	−	−	−	+	−	−	
TF_4	*Mean*	1.482E+01	1.346E+01	4.869E+00	2.033E+01	1.016E+01	6.746E+00	8.187E+00	9.602E+00	**2.416E+00**
	StD.	1.121E+01	9.606E+00	4.984E+00	1.166E+01	3.727E−01	1.724E+00	6.155E+00	1.177E+01	1.128E+00
	h	+	+	+	+	+	+	+	+	
TF_5	*Mean*	5.758E+02	2.913E+02	4.137E+02	6.077E+02	6.190E+02	5.136E+02	3.336E+02	3.321E+02	**1.033E+02**
	StD.	9.364E+01	8.951E+02	6.487E+02	4.651E+02	1.463E+02	4.576E+02	1.755E+02	2.407E+02	2.240E+02
	h	+	+	+	+	+	+	+	+	
TF_6	*Mean*	4.693E+03	5.831E+03	1.134E+03	4.026E+03	1.719E+03	**7.113E+02**	1.782E+03	1.434E+03	**4.459E+02**
	StD.	6.412E+03	1.041E+04	5.745E+03	9.113E+03	3.963E+02	7.152E+02	1.378E+04	1.802E+03	1.425E+03
	h	+	+	+	+	+	+	+	+	
TF_7	*Mean*	5.556E+00	3.597E+00	1.338E+00	1.505E+00	1.530E+00	**8.868E−01**	1.964E+00	1.482E+00	**4.503E−01**
	StD.	2.116E+00	4.304E−01	1.992E−01	7.871E−01	4.243E−01	2.389E−01	4.073E−01	8.537E−01	1.496E+00
	h	+	+	+	+	+	+	+	+	
TF_8	*Mean*	6.531E+03	1.813E+03	1.272E+03	3.151E+02	4.869E+02	6.890E+02	**1.789E+02**	1.411E+03	4.891E+02
	StD.	4.977E+04	3.507E+03	2.536E+03	9.017E+01	2.876E+02	1.668E+03	6.332E+01	2.672E+03	1.122E+02
	h	+	+	+	−	−	+	−	+	
TF_9	*Mean*	1.071E+02	1.006E+02	1.002E+02	1.003E+02	1.003E+02	1.002E+02	1.004E+02	**1.002E+02**	**1.002E+02**
	StD.	1.788E+01	1.011E+00	7.993E−02	2.752E−02	1.389E−02	1.922E−01	5.410E−01	8.993E−02	2.005E−01
	h	+	+	+	+	+	+	+	+	
TF_{10}	*Mean*	1.001E+04	1.738E+03	1.681E+03	1.559E+03	9.444E+02	7.845E+02	**3.333E+02**	1.371E+03	**6.181E+02**
	StD.	6.397E+02	3.636E+03	3.248E+03	6.499E+03	1.649E+03	4.189E+02	2.428E+01	2.726E+03	4.332E+02
	h	+	+	+	+	+	+	−	+	
TF_{11}	*Mean*	3.140E+02	2.591E+02	2.152E+02	3.003E+02	**1.618E+01**	4.643E+01	**1.308E+02**	2.171E+02	2.755E+02
	StD.	2.215E+01	1.105E+02	5.622E+02	4.292E−01	5.148E+00	1.132E+02	4.489E+02	5.358E+02	6.978E+01
	h	+	−	−	+	−	−	−	−	
TF_{12}	*Mean*	1.096E+02	1.032E+02	**1.013E+02**	1.031E+02	1.025E+02	1.018E+02	1.018E+02	1.023E+02	**1.015E+02**
	StD.	4.939E+01	1.104E+00	2.766E−01	2.417E+00	1.048E−01	6.367E−01	7.689E−01	7.934E−01	1.183E+00
	h	+	+	−	+	+	+	+	+	
TF_{13}	*Mean*	4.118E+01	3.615E+01	3.116E+01	3.883E+01	3.518E+01	3.226E+01	**3.033E+01**	3.278E+01	**3.092E+01**
	StD.	1.663E+01	2.580E+00	1.095E+01	1.020E+01	2.658E+00	5.456E+00	6.901E+00	2.861E+00	7.432E−01
	h	+	+	+	+	+	+	−	+	
TF_{14}	*Mean*	6.245E+03	5.472E+03	6.193E+03	4.292E+03	2.446E+03	**2.291E+03**	3.170E+03	**2.206E+03**	2.641E+03
	StD.	4.373E+03	6.319E+03	8.617E+03	5.329E+03	4.505E+02	1.696E+03	2.891E+03	2.297E+03	3.597E+03
	h	+	+	+	+	−	−	+	−	
TF_{15}	*Mean*	1.246E+02	1.061E+02	**1.000E+02**	1.000E+02	**1.000E+02**	1.000E+02	1.000E+02	1.000E+02	**1.000E+02**
	StD.	8.428E+00	4.052E+01	0.000E+00	7.520E−14	0.000E+00	1.730E−12	2.632E−13	1.880E−13	0.000E+00
	h	+	+	=	=	=	=	=	=	
	w/t/l/	14/0/1	13/0/2	12/1/2	12/1/2	10/1/4	12/1/2	10/1/4	11/1/3	−/−/−

4.4. Comparison of VCPSO with other evolution algorithms

In this section, we compare the performance of VCPSO with other efficient evolutionary algorithms, such as the differential evolution (DE) algorithm and the covariance Matrix Adaptation Evolution Strategy algorithm(CMA-ES). DE proposed by Storn and Price [36] is a population-based stochastic optimization algorithm in continuous search space and uses the mutation, crossover, and selection operators to evolve a population. DE has been proven to be a powerful and efficient optimizer. The evolution process of DE starts from an initial population contained N individuals in the search space and each individual represents a candidate solution of the problem. Then these solutions are selected as parents to generate offspring individuals (solutions) by mutation and crossover operations. After that, the new population is taken as the current population for next evolution operations. The CMA-ES is considered as state-of-the-art in evolutionary computation and has been utilized as one of the standard tools for continuous optimization in many research areas. It is typically applied to solve unconstrained or bounded constraint optimization problems.

The performance of VCPSO is compared with classical DE (DE/rand/2/bin) [36], GBDE [44], SADE [33], BBDE [29], and CMA-ES [5] on 50-D and 100-D dimensional search space using CEC2015 test benchmark functions. In order to achieve a fair and reliable comparison, the population size is set to 100 for DE algorithms and the results reported in this section are averages and standard deviations over 25 independent runs. Each run test was allowed D^*10^4 fitness evaluations (FEs) for each objective function. Other configurations of the compared algorithms are set as the same ones in the paper they were

Table 11
Mean, and standard deviation for 30-dimension of CEC 2015 benchmark functions from TF_1 to TF_{15}.

Func.	item	GPSO	LPSO	SPSO	QPSO	CLPSO	FIPS(ring)	DMSPSO	FDRPSO	**VCPSO**
TF_1	2.721E+08	2.721E+08	3.515E+07	**2.229E+05**	4.832E+07	5.562E+06	3.999E+06	6.715E+06	9.193E+06	**1.642E+06**
	StD.	3.540E+08	5.918E+07	2.418E+05	1.711E+07	5.802E+06	2.416E+05	1.145E+07	1.387E+07	3.206E+06
	h	+	+	+	+	+	+	+	+	
TF_2	*Mean*	2.230E+10	1.906E+09	3.699E+03	1.860E+03	4.476E+03	6.287E+03	3.367E+03	5.026E+03	**1.555E+03**
	StD.	1.962E+09	2.094E+09	2.438E+04	4.709E+03	1.701E+03	1.427E+04	2.919E+03	1.330E+04	1.282E+04
	h	+	+	+	+	+	+	+	+	
TF_3	*Mean*	2.081E+01	2.083E+01	2.086E+01	2.073E+01	2.091E+01	2.096E+01	2.048E+01	2.073E+01	2.097E+01
	StD.	7.536E−03	3.578E−01	8.297E−02	1.986E−01	1.898E−02	9.903E−02	1.530E−01	1.463E−01	3.022E−02
	h	−	−	−	−	−	−	−	−	
TF_4	*Mean*	1.551E+02	1.026E+02	3.554E+01	1.602E+02	9.018E+01	1.541E+02	8.321E+01	7.089E+01	**1.848E+01**
	StD.	8.141E+01	1.285E+01	3.375E+00	5.002E+01	3.627E+00	4.958E+01	1.536E+01	7.127E+01	1.504E+00
	h	+	+	+	+	+	+	+	+	
TF_5	*Mean*	3.541E+03	3.190E+03	3.974E+03	4.343E+03	4.618E+03	6.308E+03	3.789E+03	3.334E+03	**1.308E+03**
	StD.	3.222E+02	8.319E+02	1.385E+02	2.642E+03	6.025E+02	1.509E+03	5.398E+02	1.392E+03	7.337E+02
	h	+	+	+	+	+	+	+	+	
TF_6	*Mean*	1.001E+07	1.304E+06	1.143E+05	1.032E+06	3.529E+05	4.368E+05	1.703E+05	5.746E+05	**4.349E+04**
	StD.	7.155E+07	5.563E+05	3.126E+04	2.532E+06	1.705E+05	5.049E+05	8.564E+04	1.209E+06	9.495E+04
	h	+	+	+	+	+	+	+	+	
TF_7	*Mean*	4.774E+01	2.462E+01	9.202E+00	1.248E+01	**9.097E+00**	1.244E+01	1.285E+01	1.218E+01	**9.193E+00**
	StD.	3.549E+01	3.686E+01	1.245E+00	8.675E+00	8.595E−01	4.088E+00	9.842E−02	9.873E+00	2.532E+00
	h	+	+	+	+	−	+	+	+	
TF_8	*Mean*	1.725E+06	2.361E+05	3.221E+04	3.768E+04	6.360E+04	4.731E+04	7.952E+04	6.207E+04	**2.558E+04**
	StD.	2.049E+07	4.109E+05	2.598E+04	5.303E+04	9.237E+04	9.127E+03	1.914E+05	5.840E+04	4.565E+04
	h	+	+	+	+	+	+	+	+	
TF_9	*Mean*	2.146E+02	1.286E+02	1.028E+02	1.041E+02	1.036E+02	1.028E+02	1.036E+02	1.303E+02	**1.023E+02**
	StD.	8.370E+01	5.079E+00	1.978E−01	2.498E+00	1.814E−01	8.504E−02	7.001E−02	7.222E+01	3.278E−01
	h	+	+	+	+	+	+	+	+	
TF_{10}	*Mean*	9.291E+06	5.858E+05	**5.968E+04**	3.801E+05	2.392E+05	2.309E+05	1.478E+05	3.551E+05	**7.355E+04**
	StD.	7.986E+06	1.312E+06	1.292E+05	4.427E+05	2.860E+05	1.368E+05	3.648E+05	1.089E+06	1.791E+05
	h	+	+	−	+	+	+	+	+	
TF_{11}	*Mean*	1.242E+03	1.044E+03	5.910E+02	6.091E+02	3.552E+02	**4.388E+02**	**5.981E+02**	7.934E+02	6.200E+02
	StD.	3.879E+02	2.085E+02	7.221E+01	1.651E+03	4.813E+01	8.053E+01	7.560E+02	3.898E+02	9.293E+01
	h	+	+	−	−	−	−	−	+	
TF_{12}	*Mean*	1.659E+02	1.153E+02	1.057E+02	1.107E+02	1.070E+02	1.061E+02	1.069E+02	1.083E+02	**1.052E+02**
	StD.	3.475E+01	1.197E+01	2.387E−01	8.384E+00	1.403E+00	2.913E+00	1.072E+00	6.196E+00	6.965E−01
	h	+	+	−	+	+	+	+	+	
TF_{13}	*Mean*	1.343E+02	1.189E+02	1.147E+02	1.441E+02	1.295E+02	1.266E+02	1.157E+02	1.234E+02	1.209E+02
	StD.	3.250E+00	1.056E+00	1.164E+01	3.737E+01	9.895E+00	1.554E+01	7.673E−01	3.933E−01	3.932E+00
	h	+	−	−	+	+	+	−	+	
TF_{14}	*Mean*	4.864E+04	3.854E+04	3.364E+04	4.369E+04	2.889E+04	2.725E+04	3.067E+04	2.738E+04	**2.654E+04**
	StD.	4.466E+04	4.233E+03	3.823E+03	1.207E+04	3.800E+02	2.969E+03	1.267E+03	3.525E+03	1.446E+03
	h	+	+	+	+	+	+	+	+	
TF_{15}	*Mean*	7.581E+02	1.222E+02	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**	**1.000E+02**
	StD.	3.006E+03	2.771E+01	0.000E+00	3.586E−08	1.880E−13	3.499E−10	1.594E−05	1.088E−01	0.000E+00
	h	+	+	=	=	=	=	=	+	
	w/t/l/	14/0/1	14/0/1	10/1/4	13/1/1	12/1/2	12/1/2	10/1/4	11/1/3	−/−/−

originally published. The performance of all the compared algorithms are evaluated in terms of mean (denoted by *Mean*) and standard deviation (denoted by *StD.*) of the function best mean solution error with $f(\mathbf{x}) - f(\mathbf{x}^*)$, where $\mathbf{x}^*$ is the global minimum optimum of the test function, and x is the best solution achieved by the test algorithm after a number function evaluations. The experimental results on 50-D and 100-D dimensional problems are shown in Tables 12 and 13. To get a statistical conclusion of the results, a two-tailed Wilcoxon rank sum tests with significance level = 0.05 is conducted to judge the significance of the performance between VCPSO and its competitor. The cases are labeled with " +"," =", and "−" when the performance of VCPSO is significantly better than, similar to and worse than the compared algorithm. The whole statistical comparison results are summarized as "*w*/*t*/*l*" at the last row in Tables 12 and 13. The symbol *w*/*t*/*l* indicates that VCPSO wins in the number of *w* benchmark functions, ties in the number of *t* functions and loses in the number of *l* functions. From the 50-D and 100-D results, it can be seen that CMA-ES algorithm performs the best among these algorithms in average. VCPSO performs better than classical DE, GBDE and BBDE on a majority of functions. The performance of DEs depends on two components, namely the trial vector generation strategy (i.e., mutation and crossover operators) and the control parameters (i.e., population size , scaling factor *F* and crossover probability *CR*). The best settings for the control parameters can be different for various optimization problems and the same functions with different requirements for consumption time and accuracy. Thus, it is often necessary to perform a time-consuming trial-and-error search procedure in order to search for the most appropriate strategy and to obtain the optimal results. The performance of SADE is only inferior

Table 12
Performance comparisons of VCPSO and other efficient evolution algorithms on CEC2015 functions with 50-D problems.

Func.	DE/rand/2/bin			GBDE			BBDE		
	Mean	*StD.*	*h*	*Mean*	*StD.*	*h*	*Mean*	*StD.*	*h*
TF_1	2.74E+07	5.65E+06	+	1.08E+07	1.26E+07	+	1.62E+07	3.41E+06	+
TF_2	3.48E+05	1.68E+05	+	7.01E+03	8.38E+03	+	1.51E+03	1.05E+03	−
TF_3	2.06E+01	4.81E−02	−	2.11E+01	5.03E−02	+	2.07E+01	2.97E−02	−
TF_4	2.36E+02	1.16E+01	+	9.88E+01	5.77E+01	−	1.84E+02	1.45E+01	+
TF_5	7.79E+03	5.66E+02	+	8.70E+03	2.98E+03	+	7.72E+03	3.56E+02	+
TF_6	4.93E+04	1.71E+04	−	2.06E+06	7.79E+05	+	2.63E+06	9.69E+05	+
TF_7	4.02E+01	6.92E+00	−	4.73E+01	3.02E+01	+	2.50E+01	1.14E+01	−
TF_8	2.95E+03	3.18E+02	−	2.29E+06	1.07E+06	+	1.95E+06	6.17E+05	+
TF_9	1.07E+02	4.89E−01	+	1.05E+02	2.48E−01	+	1.05E+02	2.59E−01	+
TF_{10}	3.90E+03	2.69E+02	−	4.64E+05	3.39E+05	+	5.75E+05	2.16E+05	+
TF_{11}	2.51E+03	2.03E+02	+	1.18E+03	1.87E+02	+	5.15E+03	6.04E+02	+
TF_{12}	1.12E+02	6.93E−01	+	1.10E+02	8.08E−01	+	1.10E+02	3.45E−01	+
TF_{13}	2.28E+02	6.03E+00	+	2.26E+02	3.87E+00	+	2.14E+02	4.43E+00	−
TF_{14}	7.60E+04	5.16E+03	+	6.59E+04	4.77E+03	−	5.58E+04	1.12E+04	−
TF_{15}	1.01E+02	1.16E−01	=	1.00E+02	1.14E−13	=	1.00E+02	0.00E+00	=
w/t/l	9 / 1 / 5			12 / 1 / 2			9 / 1 /5		
Func.	SADE			CMA-ES			VCPSO		
	Mean	*StD.*	*h*	*Mean*	*StD.*	*h*	*Mean*	*StD.*	*h*
TF_1	1.24E+06	2.86E+05	−	1.14E−14	5.68E−15	−	1.74E+06	5.48E+05	=
TF_2	1.16E+03	1.44E+03	−	1.71E−14	1.39E−14	−	3.21E+03	1.01E+02	=
TF_3	2.07E+01	5.46E−02	−	2.00E+01	7.21E−06	−	2.09E+01	6.23E−03	=
TF_4	1.64E+02	2.59E+01	+	2.06E+01	6.61E+00	−	1.00E+02	2.26E+01	=
TF_5	7.22E+03	2.71E+02	−	4.00E+03	7.55E+02	−	7.63E+03	1.27E+03	=
TF_6	1.20E+06	1.95E+06	+	2.45E+03	3.23E+02	−	5.68E+04	6.24E+03	=
TF_7	4.42E+01	7.61E+00	−	4.60E+01	9.37E−01	+	4.51E+01	4.06E+00	=
TF_8	3.80E+05	9.24E+04	+	1.60E+03	3.87E+02	−	1.46E+05	1.56E+05	=
TF_9	1.05E+02	2.40E−01	+	1.04E+02	1.17E−01	=	1.04E+02	4.93E−02	=
TF_{10}	1.48E+05	3.71E+04	−	2.15E+03	3.78E+02	−	2.63E+05	4.58E+04	=
TF_{11}	7.10E+03	8.51E+02	+	3.78E+02	4.07E+01	−	6.21E+02	1.15E+01	=
TF_{12}	1.10E+02	4.26E−01	+	1.06E+02	4.38E−01	−	1.09E+02	9.53E−02	=
TF_{13}	2.24E+02	3.83E+00	=	1.84E+02	1.05E+01	−	2.24E+02	7.44E−01	=
TF_{14}	6.76E+04	3.71E+03	−	6.51E+04	7.27E+03	−	7.15E+04	4.53E+02	=
TF_{15}	1.00E+02	1.13E−13	=	1.00E+02	9.10E−14	=	1.00E+02	1.91E−14	=
w/t/l	6/2/7			1/2/12			0/15/0		

to CMA-ES. Similar to quasi-Newton methods, the CMA-ES is a second order approach estimating a positive definite matrix within an iterative procedure, thus this makes the method feasible on non-separable and/or badly conditioned problems.

4.5. Time complexity analysis

The efficiency of a given algorithm can be measured by the order of its time complexity. In this subsection, the time complexity of compared algorithms included the PSO algorithms and DE algorithms are compared. PSO updates the position by updating two vectors, namely, the position vector $\mathbf{x}$ and velocity vector $\mathbf{v}$. DE algorithms updates the solutions with three genetic operations such as the mutant operation, crossover operation and greedy selection. For PSOs, we analyze the complexity in terms of the process of initialization, evaluation, update and the total. For DE algorithms, they are compared with the items of initialization, crossover, mutant, greedy selection and total. The compared results are given in Tables 14, and 15, where *N, D, n* represent the population size, the problem's dimension, and number of neighbors respectively. For CMA-ES algorithm, a principle limitation of CMA-ES results from the degrees of freedom, $(n^2+n)/2$ in the covariance matrix. The full learning task scales roughly with D^2 . A second limitation lies in the internal computational complexity, there are several steps and the complexity of this step becomes D^2 per generation. In summary, several steps in the CMA algorithm have a computational complexity of $O(D^2)$. VCPSO partitions the full dimension into several segments and updates each segment with one operator (or called update formula). The significant program difference between VCPSO and conventional PSO is the additional operations included the vector partition and operators assignment. VCPSO consumes a bit more memory to store the partition information, but it does not increase the order of complexity of PSO algorithm and hence has the overall complexity $O(ND)$.

4.6. Discussions

According to comprehensive comparisons between VCPSO and other evolutionary algorithms, VCPSO performs better in global optimization. Experimental results confirmed that the proposed algorithm performs better or competed with other

Table 13
Performance comparisons of VCPSO and other efficient evolution algorithms on CEC2015 functions with 100-D problems.

Func.	DE/rand/2/bin			GBDE			BBDE		
	Mean	*StD.*	*h*	*Mean*	*StD.*	*h*	*Mean*	*StD.*	*h*
TF_1	1.30E+08	1.55E+07	+	1.19E+07	1.32E+06	−	1.42E+08	1.51E+07	+
TF_2	4.92E+03	1.61E+03	+	2.60E+03	2.66E+03	+	8.97E+02	9.00E+02	−
TF_3	2.09E+01	1.50E−02	−	2.13E+01	3.41E−02	=	2.11E+01	2.56E−02	−
TF_4	5.97E+02	3.16E+01	+	3.28E+02	4.78E+01	+	7.31E+02	1.50E+01	+
TF_5	1.98E+04	1.44E+03	−	2.67E+04	2.59E+03	−	2.51E+04	2.22E+02	−
TF_6	1.18E+07	2.11E+06	+	6.30E+06	6.19E+06	+	1.67E+07	1.66E+06	+
TF_7	1.38E+02	6.60E+00	−	1.44E+02	2.87E+01	+	1.37E+02	2.42E+01	−
TF_8	7.81E+05	2.68E+05	−	7.95E+06	7.06E+06	+	8.50E+06	2.26E+06	+
TF_9	1.13E+02	8.41E−01	+	1.10E+02	3.51E−01	+	1.08E+02	1.11E−01	+
TF_{10}	1.27E+04	1.64E+03	−	3.28E+05	1.16E+05	+	9.20E+05	1.28E+05	+
TF_{11}	2.81E+03	9.68E+01	+	1.94E+03	3.00E+02	+	2.16E+03	1.89E+02	+
TF_{12}	1.20E+02	9.12E−01	=	1.21E+02	1.09E+00	+	1.18E+02	3.40E−01	−
TF_{13}	4.93E+02	2.93E+00	+	4.72E+02	6.29E+00	−	4.59E+02	6.34E+00	−
TF_{14}	1.95E+05	5.58E+03	+	1.48E+05	7.50E+03	+	1.37E+05	1.66E+03	−
TF_{15}	1.01E+02	1.67E−01	+	1.00E+02	1.47E−09	=	1.00E+02	1.12E−13	=
w/*t*/*l*	9 / 1 / 5			10 / 2 / 3			7 / 1 / 6		
Func.	SADE			CMA-ES			VCPSO		
	Mean	*StD.*	*h*	*Mean*	*StD.*	*h*	*Mean*	*StD.*	*h*
TF_1	1.21E+06	9.08E+05	−	1.42E−14	0.00E+00	−	6.06E+06	1.45E+05	=
TF_2	9.71E+01	1.92E+01	−	2.84E−14	0.00E+00	−	2.51E+03	1.88E+03	=
TF_3	2.10E+01	1.58E−01	−	2.00E+01	5.15E−07	−	2.13E+01	2.27E−05	=
TF_4	5.37E+02	5.03E+01	+	5.32E+01	5.85E+00	−	2.76E+02	5.96E+01	=
TF_5	2.09E+04	9.67E+02	−	1.14E+04	6.58E+02	−	2.97E+04	1.84E+02	=
TF_6	4.82E+06	9.11E+06	−	5.93E+03	7.91E+02	−	5.07E+06	8.18E+05	=
TF_7	1.37E+02	1.47E+01	−	1.30E+02	1.99E+01	−	1.42E+02	1.01E+01	=
TF_8	1.56E+06	2.82E+06	+	3.23E+03	1.28E+02	−	9.03E+05	4.39E+05	=
TF_9	1.09E+02	3.43E−01	+	1.06E+02	2.59E−01	=	1.07E+02	1.76E−01	=
TF_{10}	1.56E+04	1.26E+04	−	4.62E+03	3.91E+02	−	2.78E+05	2.40E+05	=
TF_{11}	1.85E+03	7.84E+02	+	6.36E+02	9.93E+01	−	1.78E+03	1.49E+02	=
TF_{12}	1.19E+02	2.51E−01	−	1.13E+02	1.30E−01	−	1.20E+02	3.81E−02	=
TF_{13}	4.74E+02	2.77E+00	=	3.91E+02	1.10E+01	−	4.74E+02	8.41E−01	=
TF_{14}	1.38E+05	2.25E+03	−	1.09E+05	4.46E+01	−	1.44E+05	5.44E+03	=
TF_{15}	1.00E+02	9.10E−14	=	1.00E+02	0.00E+00	=	1.00E+02	0.00E+00	=
w/*t*/*l*	4 / 2 / 9			0 / 2 / 13			0 / 15 / 0		

Table 14
The order of complexity of PSO optimizers.

Algorithm	Order of time complexity			
	Initialize	Evaluate	Update	Overall
GPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
LPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
SPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
QPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
CLPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
FIPS	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
DMSPSO	$O(ND)$	$O(ND + NnD)$	$O(ND + NnD)$	$O(NnD)$
FDRPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$
VCPSO	$O(ND)$	$O(ND)$	$O(ND)$	$O(ND)$

Table 15
The order of computing complexity of other evolutionary optimizers.

Algorithm		Order of time complexity			
	Initialize	Mutation	Crossover	Selection	Overall
DE	$O(ND)$	$O(ND)$	$O(ND)$	$O(N)$	$O(ND)$
BBDE	$O(ND)$	$O(ND)$	$O(ND)$	$O(N)$	$O(ND)$
GBDE	$O(ND)$	$O(ND)$	$O(ND)$	$O(N)$	$O(ND)$
SADE	$O(ND)$	$O((ND)$	$O(ND)$	$O(N)$	$O(ND)$

efficient algorithms in terms of search accuracy on more than a dozen benchmark functions. The operators designed aimed to enrich the population diversity and weaken the oscillation phenomenon through changes in the dimension. In VCPSO, the full dimensions of each particle were randomly partitioned into several segments, and each segment was randomly assigned one of the six operators, i.e., the increasing operator, decreasing operator, hill operator or the lake operator, centralized operator, and decentralized operator. During the evolution process, a particles position can be changed by the motions driven by these operators. F_8–F_{15}, F_{20}–F_{28} and TF_1–TF_{15} are complex problems, only the population diversity can prevent premature convergence and reach the global optimum. Most PSO variants get trapped in local optima on these functions, while VCPSO still performs well on these problems.

The centralized operator is designed based on the resultant vector of the top ranked particles. Generally, a good resultant vector can move the swarm to a promising region near the global optimum. In contrast, a bad resultant vector may lead to premature convergence or traps the swarm into local optimum. VCPSO particles enrich their own searching experiences by using information from their elite neighbors to enhance their global searching abilities. In addition, to prevent fixed neighbor learning stagnation to some degree, the decentralize operator was designed. It reserved the local optimal sub-dimensions in the swarm. Experiments showed that VCPSO displays a better or comparable performance compared to other algorithms in terms of mean best solution accuracy on most complex unimodal and multimodal problems. This can be attributed to the hybrid vector-wise updating strategy that transforms from full dimensional learning to sub-dimensional vector learning. With the novel updating rule, different segments may learn from different exemplars. This strategy in some degree enhanced the population diversity and generated more candidate solutions. Overall, the proposed vector coevolving optimization algorithm significantly improved performance over the original PSO and can be competed with most of its variants when solving complex problems.

5. Conclusions

In this paper, we proposed a vector coevolving particle swarm optimization algorithm named VCPSO. It is completely different from the traditional PSO and its variants. It partitioned the full dimension of a particle into several segments randomly and then optimized each segment by one of the designed operators independently. These operators co-evolved with each other and enhanced the global and local search abilities. Comprehensive experimental tests have been performed on thirty-three benchmark functions which contain the mostly common used benchmark functions and CEC benchmark functions with unimodal, multimodal, rotated, shifted, hybrid and composition problems. Numerical experiments and statistical results show that VCPSO is better than, or at least comparable to PSO variants and some efficient DE optimizers when considering the quality of the solutions on a suite set of benchmark problems. However, the proposed algorithm did exist the phenomenon of trapping into local optimum or perform not efficient to overcome the problems with potential existence of dependency between variables especially in large optimization dimensions. The dual randomization mechanism with vector partition and operator assignment in some degree made it possible to improve the search quality and comprehensive experimental results showed that the proposed algorithm displays a better or comparable performance compared to the conventional PSO algorithm, DE algorithm and some of the state-of-the-art PSO variants in terms of solution accuracy for those variables dependency problems. In fact the vector partition technique is the main influencing factor in solving the non-separable problems. In future, we will make a deep investigate of the dimension partition or dimension recombination to further improve the performance of VCPSO on solving the complex non-separable problems.

Acknowledgment

This work was supported by funds from the National Natural Science Foundation of China under Grant No. 61572230, No. 61173078, No. 61573166. The authors would like to thank the anonymous reviewers for providing valuable comments that greatly helped us improve the contents of this paper.

References

[1] B. Alatas, E. Akin, A.B. Ozer, Chaos embedded particle swarm optimization algorithms, Chaos Solitons Fractals 40 (4) (2009) 1715–1734.
[2] A. Auger, N. Hansen, J.P. Zerpa, R. Ros, M. Schoenauer, Experimental Comparisons of Derivative Free Optimization Algorithms, in: J. Vahrenhold (Ed.), Experimental Algorithms, vol. 5526 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2009, pp. 3–15.
[3] G.Q. Bao, K.F. Mao, Particle Swarm Optimization Algorithm with Asymmetric Time Varying Acceleration Coefficients, in: International Conference on Robotics and Biomimetics, 2009, pp. 2134–2139.
[4] S. Baskar, P.N. Suganthan, A Novel Concurrent Particle Swarm Optimization, in: Evolutionary Computation, 2004. CEC2004. Congress on, vol. 1, IEEE, 2004, pp. 792–796.
[5] H.G. Beyer, B. Sendhoff, Covariance Matrix Adaptation Revisited the Cmsa Evolution Strategy, in: Parallel Problem Solving From Nature - PPSN X, International Conference Dortmund, Germany, 2008, pp. 123–132. September 13–17, 2008, Proceedings.
[6] D. Bratton, J. Kennedy, Defining a Standard for Particle Swarm Optimization, in: Swarm Intelligence Symposium, 2007. SIS 2007, IEEE, 2007, pp. 120–127.
[7] W.N. Chen, J. Zhang, Y. Lin, N. Chen, Z.H. Zhan, S.H. Chung, Y. Li, Y.H. Shi, P.s. o. w. a.a.l.a. challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.
[8] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, Cybern., IEEE Trans. 45 (2) (2015) 191–204.
[9] R. Cheng, C. Sun, Y. Jin, A Multi-swarm Evolutionary Framework Based on a Feedback Mechanism, in: Evolutionary Computation (CEC), 2013 IEEE Congress on, IEEE, 2013, pp. 718–724.
[10] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, Evol. Comput. IEEE Trans. 6 (1) (2002) 58–73.

[11] J. Ding, J. Liu, K.R. Chowdhury, W. Zhang, Q. Hu, J. Lei, A particle swarm optimization using local stochastic search and enhancing diversity for continuous optimization, Neurocomputing 137 (137) (2014) 261–267.
[12] M. Eslami, H. Shareef, M. Khajehzadeh, A. Mohamed, A survey of the state of the art in particle swarm optimization, Res. J. Appl. Sci., Eng. Technol. 4 (9) (2012) 1181–1197.
[13] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: acase study on the cec' 2005 special session onreal parameter optimization, J. Heuristics 15 (6) (2008) 617–644.
[14] W. Han, P. Yang, H. Ren, J. Sun, Comparison Study of Several Kinds of Inertia Weights for Pso, in: Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on, volume 1, IEEE, 2010, pp. 280–284.
[15] M. Hu, T.-F. Wu, J.D. Weir, An adaptive particle swarm optimization with multiple adaptive methods, Evol. Comput., IEEE Trans. 17 (5) (2013) 705–720.
[16] J. Kennedy, Small Worlds and Mega-minds: Effects of Neighborhood Topology on Particle Swarm Performance, in: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3, 1999, pp. 19–38.
[17] J. Kennedy, Bare Bones Particle Swarms, in: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE, 2003, pp. 80–87.
[18] J. Kennedy, R. Eberhart, Particle Swarm Optimization, in: IEEE International Conference on Neural Networks, 1995. Proceedings, volume 4, 1995, pp. 1942–1948.
[19] J. Kennedy, R. Mendes, Population structure and particle swarm performance, in: Evolutionary Computation, 2002. CEC'02. Proceedings 2002, IEEE, volume 4, 2002, pp. 1671–1676.
[20] C. Li, S. Yang, T.T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, Syst., Man, Cybern., Part B, IEEE Trans. 42 (3) (2012) 627–646.
[21] X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, Evol. Comput., IEEE Trans. 16 (2) (2012) 210–224.
[22] J. Liang, A. Qin, P. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, Evol. Comput., IEEE Trans. 10 (3) (2006) 281–295.
[23] J. Liang, P. Suganthan, Dynamic Multi-swarm Particle Swarm Optimizer, in: Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE, 2005, pp. 124–129.
[24] J. Liang, B. Qu, P. Suganthan, Q. Chen, Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Learning-based Real-Parameter Single Objective Optimization, Zhengzhou University and Nanyang Technological University, 2014.
[25] M. Mahmoodabadi, Z.S. Mottaghi, A. Bagheri, Hepso: high exploration particle swarm optimization, Inf. Sci. 273 (2014) 101–111.
[26] Y. Marinakis, M. Marinaki, A Hybridized Particle Swarm Optimization with Expanding Neighborhood Topology for the Feature Selection Problem, in: Hybrid Metaheuristics, Springer, 2013, pp. 37–51.
[27] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, Evol. Comput., IEEE Trans. 8 (3) (2004) 204–210.
[28] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Appl. Soft Comput. 11 (4) (2011) 3658–3670.
[29] M.G.H. Omran, A.P. Engelbrecht, A. Salman, Bare bones differential evolution, Eur. J. Oper. Res. 196 (1) (2009) 128–139.
[30] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, Evol. Comput., IEEE Trans. 10 (4) (2006) 440–458.
[31] Y.V. Pehlivanoglu, A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks, Evol. Comput. IEEE Trans. 17 (3) (2013) 436–452.
[32] T. Peram, K. Veeramachaneni, C. Mohan, Fitness-Distance-Ratio Based Particle Swarm Optimization, in: Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, 2003, pp. 174–181.
[33] A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Trans. Evol. Comput. 13 (2) (2009) 398–417.
[34] Y. Shang, Y. Qiu, A note on the extended rosenbrock function, Evol. Comput. 14 (1) (2006) 119–126.
[35] Y. Shi, R. Eberhart, A Modified Particle Swarm Optimizer, in: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, IEEE, 1998, pp. 69–73.
[36] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[37] P.N. Suganthan, Particle Swarm Optimiser with Neighbourhood Operator, in: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3, IEEE, 1999.
[38] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization, Nanyang Technological University.
[39] C. Sun, J. Zeng, J. Pan, S. Xue, Y. Jin, A new fitness estimation strategy for particle swarm optimization, Inf. Sci. 221 (2013) 355–370.
[40] J. Sun, B. Feng, W. Xu, Particle Swarm Optimization with Particles Having Quantum Behavior, in: Congress on Evolutionary Computation, 2004.
[41] J. Sun, W. Xu, W. Fang, A Diversity-Guided Quantum-behaved Particle Swarm Optimization Algorithm, in: Simulated Evolution and Learning, International Conference, Seal 2006, Hefei, China, October 15–18, 2006, Proceedings, 2006, pp. 497–504.
[42] M. Tanweer, S. Suresh, N. Sundararajan, Self regulating particle swarm optimization algorithm, Inf. Sci. 294 (2015) 182–202.
[43] F.V.d. Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, Evol. Comput., IEEE Trans. 8 (3) (2004) 225–239.
[44] H. Wang, S. Rahnamayan, H. Sun, M.G. Omran, Gaussian bare-bones differential evolution, IEEE Trans. Cybern. 43 (2) (2013) 634–647.
[45] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. Int. J. 223 (2) (2013) 119–135.
[46] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, Appl. Soft Comput. 48 (2016) 584–596.
[47] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, Q. Tian, Self-adaptive learning based particle swarm optimization, Inf. Sci. 181 (20) (2011) 4515–4538.
[48] W. Xu, Z. Geng, Q. Zhu, X. Gu, A piecewise linear chaotic map and sequential quadratic programming based robust hybrid particle swarm optimization, Inf. Sci. 218 (1) (2013) 85–102.
[49] Z.-H. Zhan, J. Zhang, Y. Li, Y.h. Shi, Orthogonal learning particle swarm optimization, Evol. Comput., IEEE Trans. 15 (6) (2011) 832–847.
[50] X. Zhao, Z. Liu, X. Yang, A multi-swarm cooperative multistage perturbation guiding particle swarm optimizer, Appl. Soft. Comput. 22 (2014) 77–93.