An improved dynamic cooperative random drift particle swarm optimization algorithm based on search history decision

Ji Zhao¹,², Yi Fu¹ and Juan Mei¹

Abstract
A novel dynamic cooperative random drift particle swarm optimization algorithm based on entire search history decision (CRDPSO) is reported. At each iteration, the positions and the fitness values of the evaluated solutions in the algorithm are stored by a binary space partitioning tree structure archive, which leads to a fast fitness function approximation. The mutation is adaptive and parameter-less because of the fitness function approximation enhancing the mutation strategy. The dynamic cooperation between the particles by using the context vector increases the population diversity helps to improve the search ability of the swarm and cooperatively searches for the global optimum. The performance of CRDPSO is tested on standard benchmark problems including multimodal and unimodal functions. The empirical results show that CRDPSO outperforms other well-known stochastic optimization methods.

Keywords
Random drift particle swarm optimization, dynamic cooperative evolution, search history, adaptive mutate, binary space partitioning

Received 13 January 2019; revised 6 June 2020; accepted 26 October 2020

Introduction
With the development of science and technology, many real-world engineering optimization problems become increasingly complex. Better and more efficient optimization algorithms are urgently needed. In recent years population-based optimization techniques motivated by evolution as seen in nature such as Genetic Algorithm (GA), Evolutionary Programming (EP), Evolution Strategies (ES) and Particle Swarm Optimization (PSO) have been successfully applied to solve various fields of science and technology. Particle Swarm Optimization (PSO) is a metaheuristic optimization algorithm introduced by Kennedy and Eberhart¹ to solve optimization problems inspired by the collaborative manner of social animals searching for food, such as birds flocking and fish schooling. In PSO, each volume-less individual in the swarm is called a particle, and exchange of information between particles instead of the crossover and mutation operations used in other evolutionary algorithms. PSO has been widely used to solving a number of applications owing to its simplicity and efficiency.

However, the PSO has been demonstrated that it is not a global optimization algorithm.² Over the past decades, numerous attempts have been proposed to improve the performance of the standard PSO on both exploitative and exploratory capability. To improve the global search ability of PSO, a variant of the PSO algorithm named random drift particle swarm optimization (RDPSO) was proposed by Sun³ which is inspired by the free electron model of metal conductors in an external electric field.⁴ RDPSO has been

¹School of IoT Engineering, WuXi City College of Vocational Technology, WuXi, China
²School of IoT Engineering, Jiangnan University, Wuxi, China

Corresponding author:
Ji Zhao, School of IoT Engineering, WuXi City College of Vocational Technology, Wuxi, China.
Email: queenji97@sohu.com
successfully used to obtain effective parameter estimation for complex biochemical dynamic systems and obtained better quality solutions than most global optimization methods for solving inverse problems. RDPSO algorithm can also be used as a general tool for optimization problems that widely exist in the real world. In literature, RDPSO is used to obtain the expected optimal cost of the thermal power and ESS operation under different scenarios, showing its better performance than the PSO algorithm in terms of running time, the costs of each scenario and the target expected cost. Like the PSO algorithm, RDPSO also faces premature convergence and falls into local search optimization problems, which lead to huge performance loss and suboptimal solutions. An improved version of the random drift particle swarm optimization (RDPSO) algorithm was proposed for solving the economic dispatch problem. The algorithm integrates a self-adaption mechanism and a crossover operation as well as a greedy selection process, with the mean best position of the particles replaced by the personal best position of each particle in the velocity updating equation.

In this work, we integrate RDPSO and the search history with dynamic cooperation shown to be efficient increase the diversity of particles and further improve RDPSO’s performance on complex optimization problems. Evolutionary search history has been used in the form of memory acted as adaptive guide search strategies. However, only partial search history was used in these researches. That means a portion of the information obtained from the search history is retained, and the rest is discarded. The search history includes the operations performed, the location and the fitness values of the evaluated solutions, which are extremely useful information for improving the performance of the Swarm Intelligent Algorithms (SI). Especially, it can maintain the diversity of the swarm, guide the search direction or suggest promising prospective search areas. Most importantly, when the same optimal value reappears again in the search history, it can warn that the search may fall into local optimum. Non-revisiting strategy is integrated with intelligent algorithms to avoid the re-evaluation of the evaluated solution candidates by using binary space partitioning (BSP). The non-revisiting genetic algorithm (NrGA) is proposed to prevent solution re-evaluation in the genetic algorithm by using search history and cNrGA is another version designed for continuous variables. In cNrGA, a parameter-less mutation operation was introduced to increase the population diversity. Therefore, cNrGA is more robust than GA and has superior performance. However, due to the limitation of sub-regions, the adaptive mutation of cNrGA cannot make the gene fully explore. The HdEA in literature improved the disadvantages of the sub-regions portioned by BSP so as to enhance the mutation strategy. Although HdEA is based on the entire previous search history, the global search ability is still not strong enough due to the lack of cooperative operation. Moreover, whatever cNrGA or HdEA has complex computation and slow convergence speed because of the evolutionary operations of the GA algorithm. In literature, the ESH-QPSO is proposed to improve the performance of QPSO by overlapped sub-region and One-Particle-Flip mutation. However, fitness values are still not utilized, and the mutation operation is lack of guidance due to randomness. Thus, an improved dynamic cooperative RDPSO algorithm is proposed, which is adaptively mutated with overlapped sub-regions based on evolutionary search history. Since all updated positions are guaranteed to be novel, faster convergence speed is expected.

The rest of the paper is organized as follows. Section 2 introduces the original PSO and RDPSO. Section 3 describes the proposed CRDPSO in detail. Section 4 elaborated the experimental environment. Section 5 presents the simulation results and analysis. Finally, conclusions are given in Section 6.

Random drift particle swarm optimization

Particle swarm optimization

The proposal of the particle swarm optimization algorithm was put forward to solve optimization problems by developing computational simulations of the movement of social animals such as flocks of birds and schools of fish. Supposed in the PSO with a swarm size, each particle is treated as a volume-less individual in the N-dimensional search space. The position vector and velocity vector of the th particle at the th iteration are represented as and respectively. The particle updates according to the equations:

\[
\begin{align*}
v_{ij}(t+1) &= w \cdot v_{ij}(t) + c_1 \cdot r_{1j}(t) \cdot [(pbest_{ij} - x_{ij}(t))] + c_2 \\
&\quad \cdot r_{2j}(t) \cdot [gbest(t) - x_{ij}(t)] \\
\end{align*}
\]

\[
\begin{align*}
x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1)
\end{align*}
\]

for , where and are acceleration coefficients indicated cognitive and social factors, respectively, and is the inertia weight used to balance the global and local search abilities. The vector is the best historical position found by particle called personal best position (the position giving the
The random drift particle swarm optimization

Trajectory analysis\(^2^0\) illustrated the fact the PSO algorithm’s convergence may be attained supposing that each particle converges to its local attractor, \(p^k_i = (p^k_{1i}, p^k_{2i}, \ldots p^k_{Ni})\) defined at the coordinates

\[
p^k_{ij} = \frac{c_1 r^k_{ij} p^k_{ij} + c_2 R^k_{ij} G^k_j}{c_1 r^k_{ij} + c_2 R^k_{ij}}, 1 \leq j \leq N
\]

which can be restated as

\[
p^k_{ij} = \phi^k_{ij} p^k_{ij} + (1 - \phi^k_{ij}), 1 \leq j \leq N, \text{ where } \\
\phi^k_{ij} = \frac{c_1 r^k_{ij}}{c_1 r^k_{ij} + c_2 R^k_{ij}}
\]

In general, the acceleration coefficients \(c_1\) and \(c_2\) are set to be the same in the PSO algorithm so that \(\phi^k_{ij}\) is a uniformly distributed random number over \([0, 1]\), i.e. \(\phi^k_{ij} \sim U(0, 1)\). In effect, as the particles are convergent to their own local attractors, their personal best positions, current position, the global best positions and local attractors all converge to one region, which leads to the convergence of PSO algorithm. The particle’s directional motion toward to local attractor is similar to an election’s drift movement in metal conductors in an external electric field.\(^2^3\) As reported by the free electron model, in addition to the directional movement originated from the electric field, the electron is also in thermal movement which seems to be random migration. The whole consequence of motion of the electron is that the electron careens towards the position of minimum potential energy. Obviously, if the location of an electron is regarded as a candidate solution and the potential energy function as the objective function of the problem, the electron’s movement is very similar to the procedure of obtaining a minimal solution of the minimization problem.

Illuminated by the above facts, the particle in the swarm of the PSO behaving like an electron that moves in a metal conductor in an external electric field. Thus, the motion of the particles is superposed by the directional movement and the thermal movement toward \(p_i^k\). Consequently, the velocity of the particles can be depicted by \(V^{k+1}_{ij} = V_{1ij}^{k+1} + V_{2ij}^{k+1}\), where \(V_{1ij}^{k+1}\) and \(V_{2ij}^{k+1}\) represents the velocities of the directional motion and thermal motion to \(p_i^k\), respectively. In the drift particle swarm optimization,\(^2^4\) \(V_{1ij}^{k+1}\) is the superposition of the social and the cognitive parts in equation (1) and the \(V_{2ij}^{k+1}\) is assumed to obey the Maxwell distribution, that is, Gaussian probability distribution. This velocity update equation seems to have some effects on the search performance of PSO, but there are some defects. First of all, the tail of the Gaussian distribution is very thin, so there is less chance of generating outliers. In consequence, the thermal movement of the particle owns less randomness which cannot significantly improve the global search ability of the particle. Secondly, the update equation of \(V_{1ij}^{k+1}\) can be represented as:

\[
V_{1ij}^{k+1} = c_1 r^k_{ij} (p^k_{ij} - X^k_{ij}) + c_2 r^k_{ij} (G^k_j - X^k_{ij})
\]  

The two random scaling factors \(r^k_{ij}\) add randomness to the movement of a particle in equation (7), which means that the position of the particle is sampled uniformly and distributed random within a super-rectangular shape around its personal best position and the global optimal position. Although this can guarantee the particle converge to its local attractor, it is unable to improve the overall search ability of particles because of the limited range of hyperrectangle, but may weaken the local search which is the responsibility of directional motion. Thus, the particle velocity decided by the combination of \(V_{1ij}^{k+1}\) and \(V_{2ij}^{k+1}\) may not able to achieve a good balance between the local search and the global search of the particle. A new method to determine \(V_{1ij}^{k+1}\) and \(V_{2ij}^{k+1}\) is proposed in literature.\(^3\)

The velocity of directional motion \(V_{1ij}^{k+1}\) can be simply expressed as

\[
V_{1ij}^{k+1} = \beta (p^k_{ij} - X^k_{ij})
\]
where $p^k_{ij}$ is determined by

$$p^k_{ij} = \phi^k_{ij} p_{best}^k_{ij} + (1 - \phi^k_{ij}) g_{best}^k, \quad \phi^k_{ij} \sim U(0,1),$$

$1 \leq j \leq N \tag{9}$

where $p_{best}^k_{ij}$ is the personal best (pbest) of a particle and $g_{best}^k$ is the global best (gbest).

As for the velocity of thermal motion $V^k_{ij}$ obeys double exponential distribution whose probability density function and probability distribution function are

$$f_{V^k_{ij}}(v) = \frac{1}{\sigma^k_{ij}} e^{-\frac{|v|}{\sigma^k_{ij}}} \quad \tag{10}$$

and

$$F_{V^k_{ij}}(v) = 1 - e^{-\frac{|v|}{\sigma^k_{ij}}} \tag{11}$$

respectively, where $v$ represents the value of the random variable $V^k_{ij}$, $\sigma^k_{ij}$ is the standard deviation of the distribution. $V^k_{ij}$ can be expressed as fellows by stochastic simulation method

$$V^k_{ij} = \frac{2}{\sigma^k_{ij}} \phi^k_{ij} \tag{12}$$

where

$$\phi^k_{ij} = \begin{cases} +ln(1/w^k_{ij}) & \text{if } s > 0.5 \\ -ln(1/w^k_{ij}) & \text{if } s \leq 0.5 \end{cases} \tag{13}$$

In equation (13) $s$ and $w^k_{ij}$ are two different random numbers uniformly distributed on $(0,1)$, i.e. $s, k \sim U(0,1)$. It can be directly proved that $V^k_{ij} = V^k_{ij}$, when $k \to \infty$, $X^k_{ij} \to p^k_{ij}$. Therefore, the express $V^k_{ij}$ in equation (8) does ensure that the particle points to $p^k_{ij}$ as an overall result. Then, the value of $\sigma^k_{ij}$ is determined by an adaptive strategy, i.e. $\sigma^k_{ij} = 2\|C^k_j - X^k_{ij}\|$, where $C^k_j = (c^k_1, c^k_2, \ldots, c^k_D)$ is defined as the mean of the pbest positions of all the particles named mean best (mbest) position as followed

$$C^k_j = (1/m) \sum_{i=1}^{m} p^k_{ij} \quad (1 \leq j \leq N) \tag{14}$$

Finally, the particles in RDPSO are updated equations as follows:

$$V^k_{ij}^{t+1} = \alpha [C^k_j - X^k_{ij}] + \beta (p^k_{ij} - X^k_{ij}) \tag{15}$$

where $\alpha$ means compression-expansion coefficient and $\beta$ is named acceleration coefficient. The pseudo-code of the standard RDPSO algorithm is showed as A1.

**Algorithm A1: Random Drift Particle Swarm**

*Input:* 1) a D-dimension minimization problem $f(.)$ 2) search space $S \subseteq \mathbb{R}^D$ 3) a particle swarm with population size $m$

*Output:* the optimal solution $\text{gbest}$

1. **Begin**
2. Initialize the current positions and velocities of a swarm with $m$ particles randomly, evaluate the fitness of each particle. Set the pbest position of each particle to be its current position and generate the gbest of the swarm according to formula (4);
3. for $t = 1$ to maximum number of iterations
4. Compute the mbest position $C^k_j$ according to formula (14)
5. for each particle $i$ in the swarm
6. Update the velocity and current position of particle $i$ according to equations (13), (15) and (16);
7. Evaluate the objective function value $f(X^k_j)$ and update pbest $p^k_{ij}$;
8. end for
9. Update the gbest $G^k$ of the swarm;
10. end for
11. end

The RDPSO algorithm is quite different from the drift particle swarm optimization algorithm (DPSO) because it can make a better balance between the local search and the global search of the particle population leading to better performance than DPSO. In the RDPSO algorithm, $\alpha$ and $\beta$ are two very important user specified parameters that can be adjusted to balance the local and global search of the particles. How to choose these parameter values to prevent particle explosion is an unresolved issue. Setting a larger value for $\alpha$ and $\beta$ means that the particle has better global search ability, and vice versa means that it has better local search ability. Thus, a good balance between the local search and the global search of the RDPSO algorithm is vital to the algorithmic performance when it is applied to solving a problem. It is recommended in literature that $\alpha$ should be set to be no greater than 1.0 and $\beta$ to be no greater than 1.5. Furthermore, the performance of the population-based technique can also be affected by the maximum number of iterations (MaxIteration) and the population size. As well as other PSO variants, the swarm size of RDPSO is suggested that should be greater
than 20. Configuration of the MaxIteration depends on the complexity of the problem. In general, the more complex the problem, the larger MaxIteration is required. And the velocity of a particle in RDPSO is still restricted in the interval \([-V_{\text{max}}, V_{\text{max}}]\).

**Dynamic cooperative RDPSO based on search history**

**Search history scheme**

Definition 1: The Sub-region of a solution Suppose a solution \(x\) is in the search space \(D\), i.e. \(x \in D\). \(D\) is partitioned into partition set \(D = \bigcup d_i\) by a binary space partitioning (BSP) tree \(T\) and each partition \(d_i\) is recorded as a leaf node of \(T\). The partition \(d_i \subseteq D\) is defined as the ‘the sub-region of \(x\)’ if \(x \in d_i\).

In the entire search history recorder scheme, the binary space (BSP) tree is used as a tree-structure archive to memorize all visited solutions and its fitness \([x, f(x)]\). During the iterative process, the search space is partitioned into a set of regions \(D = \bigcup d_i\) according to the distribution of the evaluated solutions \(\{x_i\}\). A region \(d_i\) of the search space is regarded as a leaf node of the BSP tree. Supposing a parent node \(N\) has two child nodes \(n_1\) and \(n_2\). The sub-region of \(N\) is linearly partitioned into two overlapped sub-regions by the child nodes. The corresponding partitioning of the search space is cut along the \(j^{\text{th}}\) dimension where \(j = \arg\max_{x_1} (j) - n_2(j)\). With this approach, each previous solution generated by the RDPSO with its fitness value can be stored in a node of the tree. Therefore, throughout the solution process, the BSP tree records all the solutions and analyzes the current solution by using the evolutionary search history of previous solutions. Since the construction of the tree depends on the order of solution set generated and evaluated by RDPSO, the BSP tree is a random tree whose topology varies from iteration to iteration.

The nodes in the BSP tree linearly divide adjacent sub-regions into a certain overlap with each other, that is, overlapped sub-regions. The method creates a channel in which particles could migrate from one sub-region to another with better fitness, as shown in Figure 1. The search for the overlapped sub-region of a solution \(x\) is executed as tree nodes search. The BSP tree \(T\) is initialized to consist of the root node \(D\) only representing the entire search space. The search starts with checking the root node. Every iteration the search moves downward from the root node and \(d\) is shrunk along a specified direction until the search reaches a leaf node. The procedure to find the overlapped sub-region of a solution \(x\) is summarized in algorithm A2. Assume that there are two evaluated solutions in the two-dimensional search space indicated as \(m = [0.25, 0.5]\) and \(n = [0.75, 0.5]\). The squares shown in Figure 1(a) and (b) represent the same sub-region of the parent node \(P\). Since \(m\) and \(n\) have maximum distance along \(x\) axis, the partitioning cuts along \(x\) axis. The overlapped sub-regions of \(m\) and \(n\), namely \(p_a\) and \(p_b\), are defined as the gray-filled regions shown in literature 1(a) and (b) respectively. It is worth noting that the union of \(r_a\) and \(r_b\) creates a path where particles in \(r_a\) can move to \(r_b\) by mutation, and vice versa. In addition, the sub-region contraction scheme of algorithm A2 ensures that the resultant sub-region overlaps with all its adjacent sub-regions. Consequently, the opinion of overlapping sub-regions integrated with mutation allows the particles to gradually approach their optimal solution nearby. Figure 2 gives an example of the BSP tree memorized four evaluated solutions: \(x_1, x_2, x_3\) and \(x_4\) in two dimensional. The distribution of the evaluated solutions in search space \(S = [0,1]^2\) is shown in Figure 2(a). Figure 2(b) shows the corresponding BSP tree. The archive tree consists of seven nodes. The solutions \(x_1\) and \(x_2\) are stored in node \(C\) and \(D\) respectively whilst the solutions \(x_3\) and \(x_4\) are stored in node \(E\) and \(F\) respectively.

Considering that the fitness values of the evaluated solutions are also stored in the memory archive by the BSP tree; this archive can also be regarded as the approximation \(f(x)\) of a fitness function \(f(x)\).

![Figure 1](image-url). The example of overlapping sub-regions.
Algorithm A2: Overlapped Sub-region Search

Input: 1) BSP Tree $T$, 2) solution $x \in \mathbb{R}^D$ ($D$ is dimension) and 3) search space $S$ is partitioned into a set of regions

$h = \prod \{i, u_i\} = S$

Output: the sub-region $r$ of solution $x$

1. Current_node = root node of $T$
2. While (Current_node has two child nodes $n_1$ and $n_2$ represented left and right respectively)
   3. $i = \arg\max_{i=1,2} (n_i(i) - n_2(i), i = 1, 2, \ldots, D)$
   4. if $(n_1(i) - x(i)) \leq n_2(i) - x(i))$
      5. Current_node = $n_1$
   6. else
      7. Current_node = $n_2$
   8. $l = n_1(i)$
   9. $u = n_2(i)$
10. End if
11. Loop

If the sub-region of an evaluated solution $x_i$ is $d_i$, the fitness value of an unseen solution $y_i$ inside the sub-region $d_i$ can be approximated as $y_i$, i.e., $\bar{f}(y) = \hat{f}(y) = y_i = f(x_i)$. Because the fitness of all solutions in the sub-region of a BSP tree node is approximated to the same value, i.e., $\bar{f}(a) = \bar{f}(b) = y_i = f(x_i)$ for all $a, b, x_i \in d_i$, $\hat{f}(x)$ is a step-wise function. In the course of evolution, the approximation error of $\hat{f}(x)$ decreases monotonously because estimated solutions gradually increase and recorded one by one. Hence the approximation process can be thought of as a simple incremental learning method of machine learning.

An adaptive parameter-less mutation mechanism

Definition 2: The neighborhood of a sub-region The sub-region $N$ of node $n$ is a neighborhood of the sub-region $M$ of the leaf node $m$ if $M \subseteq N$.

Each sub-region can be viewed as optimal in a certain amount neighborhood named neighborhood size that refers to the number of neighbor points (evaluated solutions) focused. The neighborhood size of a node decreases with its node depth due to the topology of the BSP tree. Thus, the neighborhood size is constituted by the tree node depth difference. If the sub-region $M$ of node $m$ is the neighborhood of the sub-region $N$ of node $n$, the neighborhood size (denoted by $L$) of $M$ related to $N$ is defined as the depth difference of $m$ and $n$. $M$ is assigned as an optimal sub-region in case the fitness value of node $n$ is the minimum within all evaluated solutions in $N$. Since $L$ is selected to be the depth of the BSP tree, it is presumed that the approximated fitness functions consist of only one optimum. All particles are imposed to move toward the best-found optimal value. If $L$ equals to zero, each sub-region is considered as a local optimum. Then the particles are randomly mutated within its own sub-region. To make the approximated fitness function locally guide the search with adaptive mutation, $L$ is set to be 2 in this study because $L$ should be slightly larger than zero but much smaller than the depth of the BSP tree.

Definition 3: Sub-region Distance Suppose the sub-regions of nodes $m$ and $n$ are $M$ and $N$ respectively, let the sub-region $X$ of node $x$ is the smallest sub-region which contains both $M$ and $N$, i.e., $M, N \subseteq X$, the definition of the sub-region distances from $M$ to $N$ is the depth difference between $m$ and $x$.

Once the optimal sub-region of the approximate fitness function is found, the direction of the sub-region $M$ pointing to its nearest optimal sub-region is taken as its mutation direction. Suppose $H$ is the optimal sub-region set in search space $D$ and the sub-regions of nodes $m$ and $n$ are $M$ and $N$ respectively. If the distance from $M$ to $N$ is the minimum amongst the distance from $M$ to all optimal sub-regions, i.e., $N = \arg\min_{A \subseteq H} \text{Distance} (M, A)$, the sub-region $N \subseteq H$ is the nearest optimal

![Figure 2. The BSP Tree of solutions.](image-url)
sub-region of the sub-region $M$. The more details about Definition 2 and 3 can refer to literature.\textsuperscript{15}

The mutation direction of the nodes in sub-region point to its nearest optimal neighborhood sub-region. In the adaptive parameter-less mutation mechanism, a particle $m$ moves along the direction of the local maximum incarmomement of approximation fitness $\hat{f}(x)$ at $m$. Suppose $\nu$ as the mutation direction of node $n$ pointing to the nearest historical optimum $m$ i.e., $\nu = m - n$. The best form of approximation fitness $\hat{f}(x)$ is a $D$-dimensional sub-region rather than a $D$-dimensional point because it can be regarded as a step function. Furthermore, the topological structure of the BSP tree describes the adjacencies of sub-regions amongst the steps. Therefore, the process to find the nearest optimum of node $m$ equates to find the nearest optimal sub-region of $m$ in $\hat{f}(x)$. A parameter $\alpha$ selected in the interval (0, 1) randomly is set as the mutation step size to balance the exploitative effect of this decreasing gradient like direction allocation, hence the mutant $m'$ of $m$ is a linear combination of the $m$ and $n$, $m' = m + \alpha n$.

Algorithm A3 shows the procedure of the adaptive mutation. Seen from A3, the mutation step size is randomly assigned according to the search history within an adaptive adjustment range, whilst its search direction is determined by the approximation fitness function from the BSP tree. Therefore, the mutation mechanism is an adaptive, parameter-less and guided mutation operator. Moreover, it naturally avoids the creation of any out-of-bound particles, which is superior to some other methods that might produce out-of-bound solutions and an additional repair operator is needed to change the solutions back to valid ones.

**Algorithm A3: Adaptive Parameter-less Mutate in CRDPSO**

Input: 1) the particle $p$ needs to mutate, 2) BSP tree $T$ stored the evaluated solution set \{$p_i$\}
Output: $p'$
1. The partitioned sub-region set $H = \bigcup h_i$ where $h_i$ is the sub-region of $p_i$
2. Search for the sub-region $h \subseteq H$ of $p$
3. Find the nearest optimal sub-region of $h$
4. $y =$ The evaluated solution inside $h$
5. $x =$ Random(0, 1)
6. $p' = xp + (1 - x)y$

**Dynamic cooperative evolution**

Concerning high-dimensional problems, stochastic search algorithms (including random drift particle swarm optimization and genetic algorithms) suffer from the “curse of dimensionality” of the scaling problems. Since the overall fitness of a particle will be affected by every dimension, even if the dimension values of some particles are very close to the corresponding dimensions of the global optimal solution, the particles still get lower fitness. The significance of cooperation operation is that it preserves “good” dimensional information within the particle swarm and can prevent potentially useful information from being discarded unnecessarily. By performing cooperation one dimension at a time, each dimension can be evaluated separately and the most useful information saved to speed up convergence. The algorithm performance based on cooperative evolution is proved to be efficient in the high dimension problems.\textsuperscript{22,23} In the dynamic cooperative evolution, each particle $p_i$ generates four particles \{$p_{i1}, \ldots, p_{i4}$\} by equation (14) to equation (16). Therefore, five individuals are obtained and then select the best individual $p_{i, b}$ according to their fitness value among five individuals. Then $p_{i, b}$ is assigned as the current local context vector for the particle $p_i$ and cooperates with the other four individuals. Each dimension of the context vector is replaced with the corresponding dimension of each of the remaining four individuals successively, and the alternative dimension is tested whether it improves the particle fitness value. The replacement value will be adopted if it does, and the resulting vector is used to substitute for the context vector. The process is repeated until all dimensions of all five measurement vectors have been evaluated, and the best arrangement is selected to form a new particle for the new generation. In order to reduce the fitness function calculations for each dimension replacement, the dynamic cooperative evolution is improved by selecting the optimal vector and collaborating with other particles. The procedure of the dynamic cooperative evolution is shown in algorithm A4.

**Algorithm A4: Dynamic Cooperative Evolution in CRDPSO**

Input: Particle swarm $P = \{p_1, p_2, \ldots, p_n\}$
Output: Particle swarm
1. for each particle
2. generate four particles using equations (14) to (16)
3. $p_i = p_{i, b} = \arg\min\{f(p_1), f(p_2), f(p_3), f(p_4)\}$
4. $p_k = p_i$
5. $f = f(p_{i, b})$
6. for each particle $p_k$, $k = 1, \ldots, 5, k \neq i, b$
7. for each dimension $j$
8. if $f(p_{k, j}) < f$
9. $p_k = p_{k, j}$
10. End if
11. End for
12. End for
13. End for
Algorithm A5: CRDPSO

Input: 1) a D-dimension minimization problem \( f(.) \) 2) search space \( S \subset R^D \) 3) a particle swarm \( P \) with population size \( n \)
Output: the optimal solution \( g_{best} \)
1. Initialize the particle swarm \( P = \{p_1, p_2, \ldots, p_n\} \) and evaluate the fitness of each particle \( F_i = F(p_i) \)
2. Set the \( g_{best} \) position of each particle to be its current position and generate the \( g_{best} \)
3. Initialize the BSP tree \( T \) which consists of root node only
4. for \( i = 1 \) to \( n \)
5. \([p_i, F(p_i)]\) insert to \( T \)
6. end for
7. While stop criteria is not met
8. for \( i = 1 \) to \( n \)
9. \( p_i \) adaptively mutate to be \( p'_i \)
10. \( \{p_1, \ldots, p_4\} \) updated from equations (14) to (16)
11. \( \{p_1, \ldots, p_5\} \) create new next generation according to the dynamic coordination mechanism
12. end for
13. for \( i = 1 \) to \( n \)
14. \([p_i, F(p_i)]\) insert to \( T \)
15. end for
16. for \( i = 1 \) to \( n \)
17. if \( F(p_i) < F(g_{best}) \)
18. \( g_{best} = p_i \)
19. end if
20. end for
21. Update global best fitness \( g_{best} \)
22. Loop

Dynamic cooperative RDPSO algorithm based on search history decision

Dynamic cooperative RDPSO algorithm based on entire search history is a real coded stochastic swarm intelligence algorithm. The whole algorithm searches for the optimum by main four parts that are swarm initialization, evolution, mutation and dynamic cooperative operation. CRDPSO algorithm focuses on selecting the optimal particle through the dynamic coordination mechanism and enhancing adaptive mutation based on the evolutionary search history. The procedure of CRDPSO is outlined in algorithm A5. For a \( D \)-dimensional minimization problem \( F(.) \) with the search space \( S \subset R^D \), a particle \( p \) in the swarm is a \( 1 \) by \( D \) real-valued vector, i.e., \( p \in R^D \). The algorithm starts from initializing the current population of \( n \) particles \( P = \{p_1, p_2, \ldots, p_n\} \) whilst the BSP tree \( T \) is initialized to be composed of a root node only. Afterward the population is evaluated and its fitness is both stored by \( T \). Each particle in \( P \) performs mutation adaptively, i.e. \( p_i \rightarrow p'_i \). Afterwards each particle \( p_i \) generates four particles \( \{p_{i0}, p_{i1}, p_{i2}, p_{i3}\} \) by RDPSO and dynamically cooperate with the corresponding mutant \( p_i' \) on each dimension, that is, the particle group of the \( i \)th particle for dynamic cooperative evolution is \( \{p_i, p_{i0}, p_{i1}, p_{i2}, p_{i3}\} \). The best permutation is selected to form a new particle for the new offspring. The fitness value of the new offspring particles needs to be recalculated and inserted into the BSP tree \( T \). The processes are repeated until the termination criterion is satisfied.

Simulation setup

Test function

A set of multimodal or unimodal benchmark functions \( F = \{f_1(x), f_2(x), \ldots, f_{10}(x)\}^{2425} \) is adopted to test for the proposed algorithm and verify its performance. The ten test functions are shown in Table 1. All the empirical studies in this paper were carried out in the MATLAB.

Algorithms setting

Besides RDPSO and CRDPSO, Differential Evolution (DE),\(^{26}\) Covariance Matrix Adaptation Evolutionary Strategies(CMA-ES),\(^{27}\) cNrGA and two versions of QPSO, i.e. CCQPSO\(^{23}\) and CLQPSO,\(^{28}\) were also tested on the test functions for performance comparison. The parameters of the CRDPSO and RDPSO in our experiments are set the same as those recommended in literature.\(^{3}\) To CLQPSO the \( \beta \) decreases linearly from 1 down to 0.5 and other factors are set as literature.\(^{28}\) To cNrGA, the cross rate is uniform crossover and sets as \( r_c = 0.5 \). For all compared algorithms the population size is set to 100 and the maximum number of iterations is set to 2000. The dimension of all the test functions is set as \( D = 30 \). Each algorithm independently runs 30 times in experiments, and the mean value of optimal solutions and standard deviations are achieved after 30 independent trials.

Simulation results

For all the test functions, the statistical values from 30 search runs of 2000 iterations by each algorithm are listed in Table 2 where “Best”, “Worst” and “Ave.” stand for the best, the worst and the mean values of optimal solutions obtain in 30 independent runs, respectively. “Std.” represents the standard deviations of the achieved optimal solutions. The value of the boldface means that the corresponding algorithm gets the optimal value amongst the test algorithm for a particular benchmark function. It can clearly be seen from Table 2, the proposed CRDPSO is the most superior to other six algorithms for all the test functions. Although RDPSO and CCQPSO have both shown outstanding performance on \( f_1, f_2 \) and \( f_6 \), CRDPSO has outperformed them on all the rest functions. It shows the
Table 1. Test functions.

| Function   | Objective Function                                                                 | X       | Global $f_{\text{min}}$ |
|------------|-----------------------------------------------------------------------------------|---------|--------------------------|
| $f_1$ Spherical model | $\sum_{i=1}^{D} x_i^2$                                                             | $[-100, 100]^D$ | 0                          |
| $f_2$ Schwefel's Problem 2.22  | $\sum_{i=1}^{D} |x_i| + \prod_{i=1}^{D} |x_i|$                         | $[-10, 10]^D$    | 0                          |
| $f_3$ Schwefel's Problem 1.2   | $\sum_{i=1}^{D} \left\{ \frac{1}{\prod_{j=1}^{i} x_j} \right\}^2$              | $[-100, 100]^D$ | 0                          |
| $f_4$ Schwefel's Problem 2.21  | $\max_{x \in [1, D]} |x|$                                                      | $[-10, 10]^D$    | 0                          |
| $f_5$ Restrigin's function     | $\sum_{i=1}^{D} x_i^2 - 10\cos(2\pi x_i) + 10$                                | $[-5.12, 5.12]^D$ | 0                          |
| $f_6$ Ackley function          | $-20\exp \left(-0.2\sqrt{\frac{D}{\sum_{i=1}^{D} x_i^2}}\right) - \exp \left(\frac{D}{\sum_{i=1}^{D} \cos(2\pi x_i)}\right) + 20 + e$ | $[-32, 32]^D$    | 0                          |
| $f_7$ Rosenbrock function      | $\sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$                      | $[-29, 31]^D$    | 0                          |
| $f_8$ Griewank function        | $\prod_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$       | $[-600, 600]^D$  | 0                          |
| $f_9$ Quartic function         | $\sum_{i=1}^{D} x_i^4 + \text{random}(0, 1)$                                   | $[-1.28, 1.28]^D$ | 0                          |
| $f_{10}$ Pathological          | $\sum_{i=1}^{D} \left(0.5 + \frac{\sin(\sqrt{|100x_i^2 + x_i^2 - 0.5|})}{1.001(x_i - 2\cos x_i + x_i^2 + 1)^{1/2}}\right)$ | $[-100, 100]^D$ | 0                          |

Table 2. The optimize results of the test function.

| Function | RDPSO | CRDPSO | CMA-ES | cNcGA | CCQPSO | DE | CLQPSO |
|----------|-------|--------|--------|-------|--------|----|--------|
| $f_1$    | Best  | 0      | 0      | 1.75897E-25 | 1.1139E-08 | 0  | 4.18904E-07 | 7.40207E-07 |
|          | Worst | 0      | 0      | 4.3217E-25  | 9.7340E-01  | 0  | 4.29766E-06 | 2.13162E-06 |
|          | Ave.  | 0      | 0      | 2.84821E-25 | 3.5001E-02  | 0  | 1.34E-06   | 1.40809E-06 |
|          | Std.  | 0      | 0      | 6.32276E-26 | 1.7732E-01  | 0  | 8.67663E-07 | 3.58137E-07 |
| $f_2$    | Best  | 0      | 0      | 1.3856E-13  | 3.1145E-06  | 0  | 0.001137E-05 | 2.9643E-05  |
|          | Worst | 0      | 0      | 2.8599E-13  | 1.4545E-02  | 0  | 0.008737E-05 | 4.6353E-05  |
|          | Ave.  | 0      | 0      | 2.3057E-13  | 1.4545E-02  | 0  | 0.002773E-05 | 4.6353E-05  |
|          | Std.  | 0      | 0      | 3.4456E-14  | 2.6594E-02  | 0  | 0.001145E-05 | 9.9336E-06  |
| $f_3$    | Best  | 9.3627E-04 | 0      | 8.2294E-24  | 4.8387E+02  | 0  | 9.2864E-12  | 39.3725E-08 |
|          | Worst | 5.6610E-02 | 0      | 3.4377E-23  | 7.991E+03   | 0  | 0.000420E-06 | 6.2943E-07  |
|          | Ave.  | 1.1162E-02 | 0      | 1.6878E-23  | 7.262E+03   | 0  | 8.8368E-05  | 9.8525E-03  |
|          | Std.  | 1.2250E-02 | 0      | 6.1177E-24  | 6.171E+02   | 0  | 3.5411E-04  | 1.9354E-03  |
| $f_4$    | Best  | 6.9065E-12 | 0      | 1.7053E-13  | 1.4496E-01  | 0  | 1.2243E-01  | 1.6449E+01  |
|          | Worst | 6.5872E-10 | 0      | 4.1211E-13  | 6.940E-01   | 0  | 5.8236E-11  | 1.7228E+00  |
|          | Ave.  | 1.5596E-10 | 0      | 2.4727E-13  | 3.5697E-01  | 0  | 1.9668E-01  | 4.3307E+01  |
|          | Std.  | 1.6442E-10 | 0      | 5.2988E-14  | 1.4813E-01  | 0  | 1.0628E-11  | 3.4863E-01  |
| $f_5$    | Best  | 9.0126E+00 | 0      | 2.6864E+01  | 3.1406E-08  | 0  | 2.9849E+00  | 1.8476E+02  |
|          | Worst | 6.5233E+01 | 0      | 8.4571E+01  | 2.0674E+00  | 0  | 1.4924E+01  | 2.2501E+02  |
|          | Ave.  | 1.7756E+01 | 0      | 4.6498E+01  | 6.7054E-01  | 0  | 9.3899E+00  | 2.0197E+02  |
|          | Std.  | 9.8049E+00 | 0      | 1.2827E+01  | 8.4448E-01  | 0  | 2.7248E+00  | 9.7150E+00  |

(continued)
The diversity in the algorithm declines in a smooth way in the later stage of the search process so that there is good balance between the global search and the local search of the algorithm, which results in an overall better performance than the other variants of PSO. Moreover, the simulation results of the empirical studies listed in Table 2 show that the performance of CRDPSO is better than the other compared algorithms. Using the search history mechanism, CRDPSO can ensure non-repeating access during the search process and rapid convergence to the local optimal. Meanwhile, it employs adaptive mutation to guide particles to search better neighborhoods and further improve the local search ability. In general, CRDPSO obtained the highest average accuracy and it also is the most stable.

Convergence curves versus iteration for the different algorithms are shown in Figure 3. The results in Table 2 show that for $f_1$, $f_2$ and $f_6$, all of RDPSO, CRDPSO, and CCQPSO finally achieve the optimal values. However, as can be seen from the convergence curves of Figure 3 (1)(2)(6) respectively, the convergence speed of CRDPSO is much better than the other two algorithms. It has converged to the global optimal value much faster because the particles can adaptive mutate once their own nearest optimal information, so the algorithm can quickly converge to the optimal value. In Figure 3, it is shown that for $f_3$, $f_5$ and $f_7$, the convergence speed and convergence accuracy of CRDPSO are far beyond the other six algorithms. The global optimal values of each benchmark were found within a short number of iterations. CRDPSO algorithm can search the optimal value much faster because the particles can adaptive mutate once their own nearest optimal neighborhood sub-region is found by using the entire evolutionary history information. Moreover, the coordination mechanisms can be used to keep the optimal information, so the algorithm can quickly converge to the optimal value. In Figure 3, it is shown that for $f_3$, $f_5$ and $f_7$, the CRDPSO was able to reached 0 that the Matalab can recognize and the logarithmic scale of 0 is minus infinity, so that the fitness value cannot be shown anymore in the figure. This also indicates that the simulation results for the objective function values obtained by CRDPSO for these test functions are

| Table 2. Continued. |
|---------------------|----------------|----------------|----------------|----------------|----------------|----------------|
| $f_6$   | RDPSO | CRDPSO | CMA-ES | cNrGA | CCQPSO | DE | CLQPSO |
| Best | 8.8818E-16 | 8.8818E-16 | 2.0000E + 01 | 5.6498E-05 | 8.8818E-16 | 1.7317E-04 | 3.5003E-04 |
| Worst | 8.8818E-16 | 8.8818E-16 | 2.1593E + 01 | 2.3660E-01 | 8.8818E-16 | 5.6041E-04 | 9.2455E-04 |
| Ave. | 8.8818E-16 | 8.8818E-16 | 2.1436E + 01 | 1.8326E-02 | 8.8818E-16 | 3.5209E-04 | 6.1720E-04 |
| Std. | 1.0029E-31 | 1.0029E-31 | 2.8602E-01 | 4.4225E-02 | 1.0029E-31 | 1.0143E-04 | 1.2824E-04 |
| $f_7$   | Best | 2.0862E-03 | 0 | 4.8427E + 00 | 1.5554E + 00 | 1.3355E + 01 | 2.2035E + 01 | 2.6395E + 01 |
| Worst | 9.2749E + 01 | 1.4777E-08 | 4.1314E + 02 | 4.3660E + 02 | 9.4946E + 01 | 2.6257E + 01 | 7.6362E + 01 |
| Ave. | 3.7207E + 01 | 5.1687E-10 | 1.9915E + 01 | 8.6977E + 01 | 3.3428E + 01 | 2.3721E + 01 | 4.1579E + 01 |
| Std. | 3.0608E + 01 | 2.6966E-09 | 2.9182E + 01 | 8.4565E + 01 | 2.8903E + 01 | 9.1871E-01 | 1.0091E + 01 |
| $f_8$   | Best | 0 | 0 | 7.4061E-04 | 0 | 9.4913E-07 | 5.8648E-06 |
| Worst | 2.9459E-02 | 0 | 1.3231E-02 | 3.0395E-02 | 3.6770E-02 | 2.6043E-05 | 3.4202E-05 |
| Ave. | 6.4840E-03 | 0 | 1.9714E-03 | 9.6984E-02 | 1.0330E-02 | 3.9483E-06 | 2.0352E-05 |
| Std. | 7.7636E-03 | 2.6966E-09 | 4.1634E-03 | 8.5924E-02 | 1.1312E-02 | 4.5272E-06 | 7.6931E-06 |
| $f_9$   | Best | 1.2928E-03 | 6.7247E-07 | 5.0404E-02 | 6.6570E-03 | 8.4983E-06 | 1.1918E-01 | 4.7632E-03 |
| Worst | 5.5234E-03 | 3.4890E-05 | 5.7714E-05 | 3.1475E-02 | 3.3212E-05 | 3.6087E-01 | 1.3281E-02 |
| Ave. | 2.9361E-03 | 9.5058E-06 | 2.5975E-01 | 1.8056E-02 | 1.8129E-05 | 2.1710E-01 | 2.1328E-02 |
| Std. | 1.0048E-03 | 9.0303E-06 | 1.2435E-01 | 7.4958E-03 | 1.1321E-02 | 4.5272E-06 | 7.6933E-06 |
| $f_{10}$ | Best | 9.2795E + 00 | 0 | 1.3231E + 01 | 2.5737E + 00 | 5.0896E + 00 | 1.1207E + 01 | 4.7307E + 00 |
| Worst | 1.1470E + 01 | 1.0629E-12 | 1.4500E + 01 | 4.5666E + 00 | 7.3723E + 00 | 1.2348E + 01 | 6.0035E + 00 |
| Ave. | 1.0337E + 01 | 5.3144E-14 | 1.3970E + 01 | 3.5592E + 01 | 6.1297E + 00 | 1.1974E + 01 | 5.5151E + 00 |
| Std. | 5.8348E-01 | 2.3767E-13 | 2.5855E-01 | 6.3615E-01 | 5.2576E-01 | 2.2182E-01 | 3.0625E-01 |
infinitely close to the theoretical values, i.e. 0. For $f_7$ in
Figure 3(7), even though the algorithm converged, it
was able to further find a new better solution in the
later stage (i.e. after 1500 iterations), implying that
CRDPSO algorithm still has a better search ability in
the final stage of the search process and has a large
chance to escape local optimum to get a better solution.
In conclusion, CRDPSO algorithm compared with
other algorithms has the best convergence performance
and optimization accuracy with a strong search ability.

In order to determine whether there was a significant
difference of the performance between CRDPSO algo-
rithm with other six algorithms in addition to the pre-
cision and convergence. Table 3 lists Wilcoxon rank
sum test results for a significance level of 0.05 between
different algorithms. Seen from Table 3 for $f_1$, $f_2$ and $f_6$,
CRDPSO, RDPSO and CCQPSO have no significant
difference. Nevertheless, for the vast majority of the
test functions, CRDPSO algorithm can get $h=1$,
shows that compared with other algorithms, the opti-
mized performance of CRDPSO has improved signifi-
cantly. It is observed that the optimal performance of
CRDPSO algorithm improved prominently with better
convergence and higher precision whether in the unim-
modal functions and multi-peak functions.

**Conclusion**

Premature convergence and diversity are the most need
to solve two problems with the RDPSO algorithm.
Therefore, an improved dynamic cooperative RDPSO
algorithm based on search history assisted (CRDPSO)
is proposed that integrates the search history scheme
and a standard random drift particle swarm optimiza-
tion (RDPSO). The proposed algorithm utilizes a
binary space partitioning tree (BSP) structure to store
the positions and the fitness values of the evaluated
solutions. The dynamic cooperation mechanism

**Table 3. The Wilcoxon rank sum test.**

| CRDPSO vs | CRDPSO vs | CRDPSO vs | CRDPSO vs | CRDPSO vs | CRDPSO vs |
|-----------|-----------|-----------|-----------|-----------|-----------|
| CCQPSO    | RDPSO     | CLQPSO    | cNrGA     | DE        | CMA_ES    |
| $p$       | $h$       | $p$       | $h$       | $p$       | $h$       |
| $f_1$     | NaN       | 0         | NaN       | 0         | 1.2118E-12| 1         |
| $f_2$     | NaN       | 0         | NaN       | 0         | 1.2118E-12| 1         |
| $f_3$     | 1.2118E-12| 1         | 1.2118E-12| 1         | 1.2118E-12| 1         |
| $f_4$     | 3.6097E-13| 1         | 1.2118E-12| 1         | 1.2118E-12| 1         |
| $f_5$     | 1.1613E-12| 1         | 1.2108E-12| 1         | 1.1001E-12| 1         |
| $f_6$     | NaN       | 0         | 1.2118E-12| 1         | 1.1001E-12| 1         |
| $f_7$     | 9.9025E-09| 1         | 4.0279E-09| 1         | 5.6027E-11| 1         |
| $f_8$     | 2.2016E-06| 1         | 5.2014E-06| 1         | 6.5095E-11| 1         |
| $f_9$     | 7.8252E-05| 1         | 4.1618E-11| 1         | 4.6181E-11| 1         |
| $f_{10}$  | 4.6908E-07| 1         | 3.1575E-12| 1         | 3.1549E-12| 1         |

Figure 3. Convergence curve of the algorithms.
between the solutions in the swarm is introduced for better use of context information of each dimension to take advantage of any new information, which improves the capability of the variation of the particles and enhances search ability helping to prevent premature convergence. Benefiting from the spatial partitioning scheme, a fast fitness function for improving the mutation strategy in CRDPSO is obtained using the archive structure. The resultant mutation is a kind of adaptive and parameter-less. The overlap between adjacent sub-regions also provides a channel for the movement of particles in adjacent areas, making it easier for particles to move from the area to a better area. In comparison to other well-known algorithms, the experimental results on ten standard testing functions indicate that the proposed algorithm has the best global optimization ability, with enhancement in both convergence speed and precision that demonstrate the effectiveness and efficiency of the CRDPSO. The improved algorithm can be proved in theory to be superiority; however, it also needs to be further tested in practical problems. Hence the CRDPSO algorithm applied to practical engineering problems becoming the focus research in the future. The next research job is to verify the CRDPSO performance of the engineering applications such as image segmentation, data clustering and parameter identification of biological pathways.

Declaration of Conflicting Interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The authors gratefully acknowledge financial support from the Qing Lan Project of Young and Middle-aged Academic Leaders in Colleges and Universities of Jiangsu Province [grant number 2019.2020]; the Qing Lan Project of Young and Middle-aged Academic Leaders of WuXi City College of Vocational Technology [grant number 2018]; and the Natural Science Foundation of the Jiangsu Higher Education Institutions of China [grant number 18KJB520046,17KJB520039].

ORCID iD
Ji Zhao https://orcid.org/0000-0003-3245-208X

References
1. Kennedy J and Eberhart RC. Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks. Perth, WA, Australia, Piscataway: IEEE, 1995, pp. 1942–1948.
2. Bergh FVD and Engelbrecht AP. A convergence proof for the particle swarm optimiser. Fundamenta Informaticae 2010; 105: 341–374.
3. Sun J, Palade V, Cai Y, et al. Biochemical systems identification by a random drift particle swarm optimization approach. BMC Bioinformatics 2014; 15: 1–17.
4. Omar M A. Elementary Solid State Physics: Principles and Applications [M]. Boston: Addison-Wesley, 1994, pp. 138–169.
5. Wu J, Zhang H, Li J, et al. Scenario based stochastic optimal operated for hybrid energy system with random drift swarm optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC), Wellington, New Zealand, 2019, pp. 1176–1181. Piscataway: IEEE.
6. Elsayed WT, Hegazy YG, El-Bages MS, et al. Improved random drift particle swarm optimization with self-adaptive mechanism for solving the power economic dispatch problem. IEEE Trans Industr Inform 2017; 13: 1017–1026.
7. Koumousis VK and Katsaras CP. A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 2006; 10: 19–28.
8. Xie XF, Zhang WJ and Yang ZL. A dissipative particle swarm optimization. In: IEEE congress on evolutionary computation (CEC), Hawaii, USA, Piscataway: IEEE, 2002, pp. 1666–1670.
9. Glover FM. TabuSearch. Kluwer Academic Publishers, Springer, 1997.
10. Reynolds RG. An overview of cultural algorithms: advances in evolutionary computation [M]. New York: McGraw Hill Press, 1999.
11. Farmer JD, Packard NH and Perelson AS. The immune system, adaptation and machine learning. Physica D 1986; 22: 187–204.
12. Yang Z and Wu A. A non-revisiting quantum-behaved particle swarm optimization based multilevel thresholding for image segmentation. Neural Comput Appl 2020; 32: 12011–12031.
13. Zhang X and Zhang X. A non-revisiting artificial bee colony algorithm for phased array synthesis. EURASIP J Wireless Commun Network 2017; 2017: 7.
14. Chow CK and Yuen SY. Continuous non-revising genetic algorithm with random search space re-partitioning and one-gene-flip mutation. In: Proceedings of the IEEE congress on evolutionary computation (CEC), Barcelona, Piscataway: IEEE, 2010, pp. 18–23.
15. Chow CK and Yuen SY. Continuous non-revising genetic algorithm with overlapped search sub-region. In: Proceedings of the IEEE congress on evolutionary computation (CEC), Brisbane, QLD, pp. 1–8. doi:10.1109/CEC.2010.5586046, 2012.
16. Yuen SY and Chow CK. A genetic algorithm that adaptively mutates and never revisits. IEEE Trans Evol Comput 2009; 13: 454–472.
17. Chow CK and Yuen SY. An evolutionary algorithm that makes decision based on the entire previous search history. IEEE Trans Evol Comput 2011; 15: 741–769.
18. Zhao J, Fu Y and Mei J. An improved QPSO algorithm based on entire search history. In: International symposium on distributed computing & applications for business engineering & science. Guiyang, Conference Publishing Services, 2015, pp. 74–77.
19. Shi YH and Eberhart R. A modified particle swarm optimizer. In: Proceedings of IEEE international conference on evolutionary computation. Anchorage, Ak, USA, Piscataway: IEEE, 1998, pp. 69–73.
20. Clerc M and Kennedy J. The particle swarm – explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 2002; 6: 58–73.
21. Sun J, Zhao J, Wu X, et al. Parameter estimation for chaotic systems with a drift particle swarm optimization method. Phys Lett A 2010; 374: 2816–2822.
22. Li Y, Xiang R, Jiao L, et al. An improved cooperative quantum-behaved particle swarm optimization. Soft Comput 2012; 16: 1061–1069.
23. Li Y, Jiao L, Shang R, et al. Dynamic-context cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation. Inform Sci 2015; 294: 408–422.
24. Koumousis VK and Katsaras CP. A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 2006; 10: 19–28.
25. Yao X, Liu Y and Lin GM. Evolutionary programming made faster. IEEE Trans Evol Comput 1999; 3: 82–102.
26. Storn R and Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Global Optimiz 1997; 11: 341–359.
27. Hansen N. The CMA evolution strategy: a tutorial. eprint arXiv:1604.00772, 2005.
28. Chen W, Zhou D, Sun J, et al. Improved quantum-behaved particle swarm optimization algorithm based on comprehensive learning strategy. Control Decision 2012; 27: 719–723.
29. Sun J, Wu X, Palade V, et al. Random drift particle swarm optimization algorithm: convergence analysis and parameter selection. Machine Learn 2015; 101: 345–376.