Mach Learn (2015) 101:345–376
DOI 10.1007/s10994-015-5522-z

Random drift particle swarm optimization algorithm: convergence analysis and parameter selection

Jun Sun[1] · Xiaojun Wu[1] · Vasile Palade[2] · Wei Fang[1] · Yuhui Shi[3]

Received: 5 February 2014 / Accepted: 25 June 2015 / Published online: 15 August 2015

Abstract The random drift particle swarm optimization (RDPSO) algorithm is a PSO variant inspired by the free electron model in metal conductors placed in an external electric field. Based on the preliminary work on the RDPSO algorithm, this paper makes systematical analyses and empirical studies of the algorithm. Firstly, the motivation of the RDPSO algorithm is presented and the design of the particle's velocity equation is described in detail. Secondly, a comprehensive analysis of the algorithm is made in order to gain a deep insight into how the RDPSO algorithm works. It involves a theoretical analysis and the simulation of the stochastic dynamical behavior of a single particle in the RDPSO algorithm. The search behavior of the algorithm itself is also investigated in detail, by analyzing the interaction among the particles. Then, some variants of the RDPSO algorithm are presented by incorporating different random velocity components with different neighborhood topologies. Finally, empirical studies of the RDPSO algorithm are performed by using a set of benchmark functions from the CEC2005 benchmark suite. Based on the theoretical analysis of the particle's

Editors: Vadim Strijov, Richard Weber, Gerhard-Wilhelm Weber, and Süreyya Ozogur Akyüz.

✉ Jun Sun
SUNJUN_WX@HOTMAIL.COM

Xiaojun Wu
WU_XIAOJUN@YAHOO.COM.CN

Vasile Palade
VASILE.PALADE@COVENTRY.AC.UK

Wei Fang
WXFANGWEI@HOTMAIL.COM

Yuhui Shi
YUHUI.SHI@XJTLU.EDU.CN

1 Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Jiangnan University, No 1800, Lihu Avenue, Wuxi 214122, Jiangsu, China

2 Department of Computing, Coventry University, Priory Street, Coventry CV1 5FB, UK

3 Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou 215123, Jiangsu, China

behavior, two methods of controlling the algorithmic parameters are employed, followed by an experimental analysis on how to select the parameter values, in order to obtain a satisfactory overall performance of the RDPSO algorithm and its variants in real-world applications. A further performance comparison between the RDPSO algorithm and other variants of PSO is made to prove the effectiveness of the RDPSO.

Keywords Evolutionary computation · Optimization · Particle swarm optimization · Random motion

1 Introduction

Optimization methods are essential in machine learning and are usually employed to find the model parameters and appropriate structures (Sra et al. 2011; Henning and Keifel 2013; Bull 2011; Sun et al. 2013). Among the various optimization methods available, random search approaches are direct search techniques that incorporate stochastic strategies into the search process to enable the algorithms to jump out of local optima with high probability, and they are widely used in machine learning (Bergstra and Bengio 2012). Metaheuristic methods are an important class of random search techniques. They are formally defined as an iterative generation-based process, which guides the search by using a subordinate heuristic and intelligently combining different concepts for exploring and exploiting the search space, the most popular methods in this class being the evolutionary algorithms (EAs) (Fortin et al. 2012; Pelikan 2012; Fournier and Teytaud 2011; Lu et al. 2011)

Particle swarm optimization (PSO) is a metaheuristic method attributed to be originally developed by Kennedy and Eberhart (1995), Eberhart and Kennedy (1995). It has been motivated by bird flocking and fish schooling mechanisms, and the swarm theory in particular. Unlike EAs, PSO has no evolution operators such as crossover and selection. The PSO algorithm performs an optimization task by iteratively improving a swarm of candidate solutions with respect to an objective (fitness) function. The candidate solutions, called particles, move through the problem space according to simple mathematical formulae describing the particles' positions and velocities. The movement of each particle is influenced by its own experiences, and is also guided towards the current best known position.

In the last decade, PSO has gained increasing popularity due to its effectiveness in performing difficult optimization tasks. The reason why PSO is attractive is that it gets better solutions, in a faster and cheaper way compared to other methods, whereas has fewer parameters to adjust. It has been successfully used in many research and application areas (Poli 2007, 2008; Veeramachaneni et al. 2012). The algorithm has also been found useful in machine learning problems (Escalante et al. 2009).

To gain insights into how the algorithm works, some researchers have theoretically analyzed the PSO algorithm. These analyses mainly aimed for the behavior of the individual particle in the PSO algorithm, which is essential to the understanding of the search mechanism of the algorithm and to parameter selection (Kennedy 1998; Ozcan and Mohan 1999; Clerc and Kennedy 2002; van den Bergh 2002; Shi and Eberhart 1998a, b; Trelea 2003; Emara and Fattah 2004; Gavi and Passino 2003; Kadirkamanathan et al. 2006; Jiang et al. 2007; Poli 2009; Bonyadi et al. 2014; Cleghorn and Engelbrecht 2014). For example, Kennedy analysed a simplified particle behavior and demonstrated different particle trajectories for a range of design choices (Kennedy 1998). Clerc and Kennedy undertook a comprehensive analysis of the particle trajectory and its stability properties (Clerc and Kennedy 2002). As

for the algorithm itself, Van den Bergh proved that the canonical PSO is not a global search algorithm, even not a local one (van den Bergh 2002; Van den Bergh and Engelbrecht 2006, 2010), by using the convergence criterion provided by Solis and Wets (1981).

In addition to the analyses mentioned above, there has been a considerable amount of work performed in improving the original version of the PSO through empirical studies. The original PSO proposed in Kennedy and Eberhart (1995) appeared to have weak local search ability, due to the slow convergence speed of the particles. It is universally known that the tradeoff between the local search (exploitation) and the global search (exploration) is vital for the performance of the algorithm. As such, the original PSO needs to accelerate the convergence speed of the particles in order to achieve a better balance between exploitation and exploration. The work in this area, which was first carried out by Shi and Eberhart, involved introducing an inertia weight into the update equation for velocities, in order to control the explosion in velocity values and partially help accelerate the convergence of individual particles (Shi and Eberhart 1998a, b). Clerc (1999) proposed another acceleration method by adding a constriction factor in the velocity update equation, in order to release the restriction on the particle's velocity during the convergence history. The acceleration techniques were shown to work well, and the above two variants of PSO have laid the foundation for further enhancement of the PSO algorithm.

In general, two types of neighborhood topologies are used when the PSO algorithm is implemented. One is known as the global best topology or global best model (essentially the star model), which is employed in the PSO with inertia weight (PSO-In) and the PSO with constriction factor (PSO-Co). In this topology model, the search of the particles is guided by the global best position as well as their personal best positions. Although the algorithm with this model is able to efficiently obtain the best approximate solutions for many problems, some researchers argued that this model may be prone to encounter premature convergence when solving harder problems. If the global best particle sticks to a local or suboptimal point, it would mislead the other particles to move towards that point. In other words, other promising search areas might be missed. This had led to the investigation of other neighborhood topologies, known as the local best (lbest) models, first studied by Eberhart and Kennedy (1995) and subsequently in depth by many other researchers (Suganthan 1999; Kennedy 1999, 2002; Liang and Suganthan 2005; Mendes et al. 2004; Parrott and Li 2006; Bratton and Kennedy 2007; Kennedy and Mendes 2002; van den Bergh and Engelbrecht 2004; Lane et al. 2008; Li 2004). The objective there was to find other possible topologies to improve the performance of the PSO algorithm. Engelbrecht (2013) carried out a comprehensive and elaborate empirical comparison of the gbest PSO and lbest PSO algorithms, on a suite of 60 benchmark boundary constrained optimization problems of varying complexities. The statistics of their experimental results show that neither of the two types of algorithms can be considered to outperform the other, not even for specific problem classes in terms of convergence, exploration ability, and solution accuracy.

Another way to possibly improve the PSO algorithm is to directly sample new positions during the search. Thus, some researchers proposed several probabilistic PSO algorithms, which simulate the particle trajectories by directly sampling according to a certain probability distribution (Kennedy 2003, 2004, 2006; Sun et al. 2012; Krohling 2004; Secrest and Lamont 2003; Richer and Blackwell 2006). The Bare Bones PSO (BBPSO) family is a typical class of probabilistic PSO algorithms (Kennedy 2003). In BBPSO, each particle does not have a velocity vector, but its new position is sampled "around" a supposedly good one, according to a certain probability distribution, such as the Gaussian distribution in the original version (Kennedy 2003). Several other new BBPSO variants used other distributions which seem to generate better results (Kennedy 2004, 2006). Recently, some researchers

employed stochastic process models, such as Markov chains, to analyse the convergence of the Bare Bone PSO (Poli and Langdon 2007; Zhang et al. 2014).

This paper is focused on the so-called random drift particle swarm optimization (RDPSO), which is inspired by the free electron model in metal conductors in an external electric field (Omar 1993). The basics of the original concept of the random drift model for PSO were sketched in our previous work (Sun et al. 2010). In the limited initial version of the algorithm (Sun et al. 2010), the velocity of the particle's drift motion is simply expressed by the summation of the cognition part and the social part in the velocity update equation of the original PSO, which is not consistent with the physical meaning of the random drift model. This paper is to use a more concise form for the drift velocity, which is more in line with the physical meaning of the model, as well as a novel strategy for determining the random velocity, and thus it presents a new and different version of the RDPSO algorithm. In Sun et al. (2014a), an RDPSO version with double exponential distribution was validated by testing the algorithm on biochemical system identification problems, while in this paper, a Gaussian distribution is employed to sample the particles' random velocities in the RDPSO algorithm. In Sun et al. (2014b), this version of RDPSO, along with two improved variants, were applied for training Hidden Markov Models for biological multiple sequence alignment, which is an important machine learning problem in bioinformatics.

This paper presents the extension of our previous work on the RDPSO algorithm. Its purpose is to gain an in-depth understanding of how the RDPSO works by making comprehensive theoretical analyses of the behavior of the individual particle in the RDPSO and the search behavior of the algorithm, and to undertake empirical studies on the issue of parameter selection for four RDPSO variants, which employ different random velocities and neighborhood topologies. Performance comparison between the RDPSO and other PSO variants is to be made by using fourteen benchmark functions from the CEC2005 benchmark suite in order to verify the effectiveness of the proposed algorithms.

The remainder of the paper is organized as follows. Section 2 describes the motivation and principle of the RDPSO algorithm. Section 3 presents the analyses of the RDPSO algorithm, and Sect. 4 presents the four RDPSO variants. Empirical studies on the parameter selection for the RDPSO algorithm and the performance comparison are provided in Sect. 5. Finally, the paper is concluded in Sect. 6.

2 Random drift particle swarm optimization (RDPSO)

2.1 Basic definitions and terminology for PSO

In a PSO with M individuals, each individual is treated as a volume-less particle in the N-dimensional space, with the current position vector and the velocity vector of particle i ($1 \leq i \leq M$) at the n^{th} iteration represented as $X_{i,n} = (X_{i,n}^1, X_{i,n}^2, \ldots, X_{i,n}^N)$ and $V_{i,n} = (V_{i,n}^1, V_{i,n}^2, \ldots, V_{i,n}^N)$. Each particle i also has the personal best (*pbest*) position vector $P_{i,n} = (P_{i,n}^1, P_{i,n}^2, \ldots, P_{i,n}^N)$ at the n^{th} iteration, which records the position giving the best fitness value (i.e. the objective function value) of the particle from the initialization to the current iteration. Besides, there is a vector $G_n = (G_n^1, G_n^2, \ldots, G_n^N)$, known as the global best (*gbest*) position, recording the position with the best fitness value found by the whole particle swarm so far. Without loss of generality, we consider the following minimization problem:

$$\text{Minimize} \quad f(X)\,, \; s.t. \quad X \in S \subseteq R^N, \tag{1}$$

where $f(X)$ is an objective function and S is the feasible space. Accordingly, $P_{i,n}$ can be found by

$$P_{i,n} = \begin{cases} X_{i,n} & \text{if} \quad f(X_{i,n}) < f(P_{i,n-1}) \\ P_{i,n-1} & \text{if} \quad f(X_{i,n}) \geq f(P_{i,n-1}) \end{cases}, \tag{2}$$

and G_n updated by

$$G_n = P_{g,n}, \text{where} \quad g = \arg \min_{1 \leq i \leq M} [f(P_{i,n})]. \tag{3}$$

In the basic PSO algorithm, the particle updates its velocity and position at the $(n+1)^{\text{th}}$ iteration according to the following equations:

$$V_{i,n+1}^j = V_{i,n}^j + c_1 r_{i,n}^j \left(P_{i,n}^j - X_{i,n}^j\right) + c_2 R_{i,n}^j \left(G_n^j - X_{i,n}^j\right), \tag{4}$$

$$X_{i,n+1}^j = X_{i,n}^j + V_{i,n+1}^j, \tag{5}$$

for $i = 1, 2, \ldots M$; $j = 1, 2 \ldots, N$, where c_1 and c_2 are known as the acceleration coefficients, and the parameters $r_{i,n}^j$ and $R_{i,n}^j$ are sequences of two different random numbers distributed uniformly on (0, 1), which is denoted by $r_{i,n}^j, R_{i,n}^j \sim U(0, 1)$. Generally, the value of $V_{i,n}^j$ is restricted within the interval $[-V_{\max}, V_{\max}]$.

2.2 The motivation of the RDPSO algorithm

It has been demonstrated that the convergence of the whole particle swarm may be achieved if each particle converges to its local focus, $p_{i,n} = (p_{i,n}^1, p_{i,n}^2, \ldots p_{i,n}^N)$, defined by the following coordinates (Clerc and Kennedy 2002):

$$p_{i,n}^j = \frac{c_1 r_{i,n}^j P_{i,n}^j + c_2 R_{i,n}^j G_n^j}{c_1 r_{i,n}^j + c_2 R_{i,n}^j}, \qquad 1 \leq j \leq N, \tag{6}$$

or

$$p_{i,n}^j = \phi_{i,n}^j P_{i,n}^j + \left(1 - \phi_{i,n}^j\right) G_n^j, \tag{7}$$

where $\phi_{i,n}^j = c_1 r_{i,n}^j / (c_1 r_{i,n}^j + c_2 R_{i,n}^j)$, with regard to the random numbers $r_{i,n}^j$ and $R_{i,n}^j$ defined in Eqs. (4), (6). The acceleration coefficients c_1 and c_2 in the original PSO are generally set to be equal, namely, $c_1 = c_2$, and thus, $\phi_{i,n}^j$ is a sequence of random numbers uniformly distributed on (0,1). As a result, Eq. (7) can be restated as

$$p_{i,n}^j = \phi_{i,n}^j P_{i,n}^j + \left(1 - \phi_{i,n}^j\right) G_n^j, \quad \varphi_{i,n}^j \sim U(0, 1). \tag{8}$$

In fact, as the particles are converging to their own local attractors, their current positions, *pbest* positions, local focuses and the *gbest* position are all converging to one point. Since $p_{i,n}$ is a random point uniformly distributed within the hyper-rectangle with $P_{i,n}$ and G_n being the two ends of its diagonal, the particle's directional movement towards $p_{i,n}$ makes the particle search around this hyper-rectangle and improves its fitness value locally. Hence, this directional movement essentially reflects the local search of the particle.

The motivation of the proposed RDPSO algorithm comes from the above trajectory analysis of the PSO and the free electron model in metal conductors placed in an external electric field (Omar 1993). According to this model, the movement of an electron is the superimposition of the thermal motion, which appears to be a random movement, and the drift motion (i.e., the directional motion) caused by the electric field. That is, the velocity of the electron

can be expressed by $V = VR + VD$, where VR and VD are called the random velocity and the drift velocity, respectively. The random motion (i.e., the thermal motion) exists even in the absence of the external electric field, while the drift motion is a directional movement in the opposite direction of the external electric field. The overall physical effect of the electron's movement is that the electron careens towards the location of the minimum potential energy. In a non-convex-shaped metal conductor in an external electric field, there may be many locations of local minimum potential energies, which the drift motion generated by the electric force may drive the electron to. If the electron only had the drift motion, it might stick into a point of local minimum potential energy, just as a local optimization method converges to a local minimum of an optimization problem. The thermal motion can make the electron more volatile and, consequently, helps the electron to escape the trap of local minimum potential energy, just as a certain random search strategy is introduced into the local search technique to lead the algorithm to search globally. Therefore, the movement of the electron is a process of minimizing its potential energy. The goal of this process is essentially to find out the minimum solution of the minimization problem, with the position of the electron represented as a candidate solution and the potential energy function as the objective function of the problem.

2.3 Description of the RDPSO algorithm

Inspired by the above facts, we assume that the particle in the RDPSO behaves like an electron moving in a metal conductor in an external electric field. The movement of the particle is thus the superposition of the thermal and the drift motions. The thermal motion implements the global search of the particle, while the drift motion mainly implements the local search. The trajectory analysis, as described in the first paragraph of this subsection, indicates that, in the canonical PSO, the particle's directional movement toward its local attractor $p_{i,n}$ reflects the local search of the particle. In the proposed RDPSO, the drift motion of the particle is also defined as the directional movement towards $p_{i,n}$. It represents the combination of the cognition part and the social part of the canonical PSO and, thus, is the main inheritance of the RDPSO algorithm from the canonical PSO algorithm. In the RDPSO algorithm, the 'inertia part' in the velocity equation of the canonical PSO is replaced by the random velocity component, which is the main difference between the RDPSO algorithm and the canonical PSO algorithm. Therefore, the velocity of the particle in the RDPSO algorithm has two components, i.e., the thermal or random component, and the drift component. Mathematically, the velocity of particle i in the jth dimension can be expressed by $V^j_{i,n+1} = VR^j_{i,n+1} + VD^j_{i,n+1}$ $(1 \le i \le M, 1 \le j \le N)$, where $VR^j_{i,n+1}$ and $VD^j_{i,n+1}$ are the random velocity component and the drift velocity component, respectively.

A further assumption is that the value of the random velocity component $VR^j_{i,n+1}$ follows the Maxwell velocity distribution law (Kittel and Kroemer 1980). Consequently, $VR^j_{i,n+1}$ essentially follows a normal distribution (i.e., Gaussian distribution) whose probability density function is given by

$$f_{VR^j_{i,n+1}}(v) = \frac{1}{\sqrt{2\pi}\sigma^j_{i,n+1}} e^{\frac{-v^2}{2\left(\sigma^j_{i,n+1}\right)^2}}, \tag{9}$$

where $\sigma^j_{i,n+1}$ is the standard deviation of the distribution. Using stochastic simulation, we can express $VR^j_{i,n+1}$ as

$$VR_{i,n+1}^{j} = \sigma_{i,n+1}^{j}\varphi_{i,n+1}^{j}, \tag{10}$$

where $\varphi_{i,n+1}^{j}$ is a random number with a standard normal distribution, i.e., $\varphi_{i,n+1}^{j} \sim N(0,1)$. $\sigma_{i,n+1}^{j}$ must be determined in order to calculate $VR_{i,n+1}^{j}$. An adaptive strategy is adopted for $\sigma_{i,n+1}^{j}$:

$$\sigma_{i,n+1}^{j} = \alpha\left|C_n^j - X_{i,n}^j\right|, \tag{11}$$

where $C_n = (C_n^1, C_n^2, \ldots, C_n^N)$ is known as the mean best (*mbest*) position defined by the mean of the *pbest* positions of all the particles, namely,

$$C_n^j = (1/M)\sum_{i=1}^{M} P_{i,n}^j, \qquad (1 \le j \le N). \tag{12}$$

Thus, Eq. (10) can be restated as

$$VR_{i,n+1}^{j} = \alpha\left|C_n^j - X_{i,n}^j\right|\varphi_{i,n+1}^{j}, \tag{13}$$

where $\alpha > 0$ is an algorithmic parameter called the thermal coefficient. In the next section, where the search behavior of individual particles and the whole swarm is analyzed, we will find that this random velocity component drives the particle away from the global best position, so it indeed reflects the global search of the particle.

The role of the drift velocity component, $VD_{i,n+1}^{j}$, is to implement the local search of the particle, which can be achieved by the directional movement toward $p_{i,n}$, as has been mentioned above. In this paper we use the following simple linear expression for $VD_{i,n+1}^{j}$:

$$VD_{i,n+1}^{j} = \beta\left(p_{i,n}^j - X_{i,n}^j\right), \tag{14}$$

where $\beta > 0$ is a deterministic constant and is another algorithmic parameter called the drift coefficient. This form of $VD_{i,n+1}^{j}$ in Eq. (14) is more concise than the one in Sun et al. (2010) and it has a clear physical meaning that it reflects the particle's directional movement towards $p_{i,n}$. In Theorem 1 in the Appendix, it is proven that, if there is only drift motion and, i.e., $V_{i,n+1}^{j} = VD_{i,n+1}^{j}$, $X_{i,n}^{j} \to p_{i,n}^{j}$ as $n \to \infty$ when $0 < \beta < 2$, meaning that the expression of $VD_{i,n+1}^{j}$ in Eq. (14) can indeed guarantee the particle's directional movement toward $p_{i,n}$ as an overall result. More specifically, if $0 < \beta < 1$, $X_{i,n}^{j}$ asymptotically converges to $p_{i,n}^{j}$, which means that the sampling space of $X_{i,n+1}$ does not cover the hyper-rectangle with $P_{i,n}$ and G_n being the two ends of its diagonal. If $\beta = 1$, $X_{i,n+1}^{j}$ is identical to $p_{i,n}^{j}$ so that the sampling space of $X_{i,n+1}$ is exactly the hyper-rectangle. If $1 < \beta < 2$, $X_{i,n}^{j}$ converges to $p_{i,n}^{j}$ in oscillation and thus the sampling space of $X_{i,n+1}$ covers the hyper-rectangle and even other neighborhoods of G_n, where points with better fitness values may exist. As such, when we select the value of β for real application of the RDPSO algorithm, it may be desirable to set $1 \le \beta < 2$ for good local search ability of the particles.

With the above specification, a novel set of update equations can be obtained for the particle of the RDPSO algorithm:

$$V_{i,n+1}^{j} = \alpha\left|C_n^j - X_{i,n}^j\right|\varphi_{i,n+1}^{j} + \beta\left(p_{i,n}^j - X_{i,n}^j\right), \tag{15}$$

$$X_{i,n+1}^{j} = X_{i,n}^{j} + V_{i,n+1}^{j}. \tag{16}$$

The procedure of the algorithm is outlined below in Algorithm 1. Like in the canonical PSO, the value of $V_{i,n}^{j}$ in the RDPSO is also restricted within the interval $[-V_{\max}, V_{\max}]$ at each iteration.

Algorithm 1: The RDPSO algorithm

```
begin
    Initialize the current positions and velocities of all the particles randomly;
    Set the personal best position of each particle to be its current position;
    Set n = 0;
    while (the termination condition=false) do
        Set n = n + 1;
        Compute the mean best position C_n and select the value of α and β
        properly;
        for i = 1 to M do
            for j = 1 to N do
                p^j_{i,n} = φ^j_{i,n} P^j_{i,n} + (1 − φ^j_{i,n}) G^j_n;
                V^j_{i,n+1} = α|X^j_{i,n} − C^j_n| φ^j_{i,n} + β(p^j_{i,n} − X^j_{i,n});
                X^j_{i,n+1} = X^j_{i,n} + V^j_{i,n+1};
            end
            Evaluate the fitness value of X_{i,n+1}, i.e. the objective function value
            f(X_{i,n});
            Update P_{i,n} and G_n according to equations (2) and (3);
        end
    end
end
```

3 Analysis of the RDPSO algorithm

3.1 Dynamical behaviour of the RDPSO particle

An analysis of the behavior of an individual particle in the RDPSO is essential to understanding how the RDPSO algorithm works and how to select the algorithmic parameters. Since the particle's velocity is the superimposition of the thermal velocity and the drift velocity, the conditions for the particle's position to converge or to be bounded are far more complex than those given in Sect. 2.3 when only the drift motion exists. In this subsection, we will carry out theoretical and empirical studies on the stochastic dynamical behavior of the particle in the RDPSO. Since each dimension of the particle's position is updated independently, we only need to consider a single particle in a one-dimensional space without loss of generality. As such, Eqs. (15) and (16) can be simplified as

$$V_{n+1} = \alpha|C - X_n|\varphi_{n+1} + \beta(p - X_n), \tag{17}$$

$$X_{n+1} = X_{n+1} + V_{n+1}, \tag{18}$$

where X_n and V_n denote the current position and the velocity of the particle respectively at the n^{th} iteration, and the local focus of the particle and the mean best position are denoted by p and C, which are treated as probabilistically bounded random variables, i.e., $P\{\sup|p| < \infty\} = 1$ and $P\{\sup|C| < \infty\} = 1$. In Eq. (17), $\{\varphi_n\}$ is a sequence of independent identically distributed random variables with $\varphi_n \sim N(0, 1)$.

Since the probability distribution of φ_n is symmetrical with respect to the ordinate, Eq. (17) has the following equivalence:

$$V_{n+1} = \alpha(X_n - C)\varphi_{n+1} - \beta(X_n - p), \tag{19}$$

that is, the probability distributions of V_{n+1} in Eqs. (17) and (19) are the same. Based on Eqs. (19) and (18), several theorems on the dynamical behavior of an individual particle in RDPSO are proved in the Appendix. As shown by Theorem 2, the particle's behavior is related to the convergence of $\rho_n = \prod_{i=1}^{n} \lambda_i$, where $\lambda_n = \alpha\varphi_n + (1 - \beta)$ subject to a normal distribution, namely, $\lambda_n \sim N(1 - \beta, \alpha^2)$. It is showed by Theorem 3 that if and only if $\Delta = E(\ln|\lambda_n|) \leq 0$, namely, the values of α and β satisfy the following relationship:

$$\Delta = \frac{1}{\sqrt{2\pi}\alpha} \int_{-\infty}^{+\infty} \ln|x| e^{-\frac{[x-(1-\beta)]^2}{2\alpha^2}} dx \leq 0, \tag{20}$$

ρ_n is probabilistically bounded and, thus, the position of the particle is probabilistically bounded too. In inequality (20), the value of Δ is an improper integral which is undefined at $x = 0$. By a Dirichlet test, the improper integral in (20) is convergent if both α and β are two finite numbers (Courant 1989).

Inequality (20) does not give any explicit constraint relation between α and β due to the difficulty in calculating the improper integral in the inequality. A sufficient condition for $\Delta < 0$ (i.e. $\lim_{n\to\infty}\rho_n = \lim_{n\to\infty}\prod_{i=1}^{n}\lambda_i = 0$) is derived in Theorems 4. It says that if the values of α and β are subject to the constraint:

$$0 < \alpha < 1, \quad 0 < \beta < 2, \tag{21}$$

$\Delta < 0$ and $\rho_n = \prod_{i=1}^{n}\lambda_i$ converges to zero, which, as a consequence, ensures the probabilistic boundedness of the particle's current position. Figure 1 traces some simulation results for the stochastic behaviour of the particle with different values of α and β, with C fixed at $X = 0.001$, p fixed at the origin and the initial position of the particle set as $X_0 = 1000$. Figure 1a–c show the results with α and β satisfying constraint (21). It can be observed that the particle's position oscillated around p and C, implying that the position is probabilistically bounded in these cases. Figure 1d–i show that the particle's position is probabilistically bounded in some cases when α and β do not satisfy constraint (21). This verifies that constraint (21) is a sufficient condition for $\Delta < 0$ or $\lim_{n\to\infty}\rho_n = 0$. At other values of α and β that do not satisfy (21), the value of $\ln|X_n - p|$ reached 700 and stopped changing after a certain number of iterations, as shown in Fig. 1j–o. In such cases, the value of $|X_n - p|$ reaches the maximum positive value that the computer can identify, so that it can be considered to have diverged to infinity.

Constraint (21) is of practical significance to the application of the RDPSO algorithm, although it does not give the necessary condition for $\Delta \leq 0$. In practice, the values of α and β can generally be selected within the intervals given by (21), for a satisfactory algorithmic performance when the algorithm is applied to real-world problems. In Sect. 4, a detailed investigation into how to select these algorithmic parameters will be undertaken by using three widely used functions and then the algorithmic performance with these parameters values will be further tested on a set of benchmark functions from the CEC2005 benchmark suite.

3.2 The RDPSO's search behavior

In the above analysis, it is assumed that each particle in the RDPSO updates its velocity and position independently, with the mean best position C and the local focus p being treated

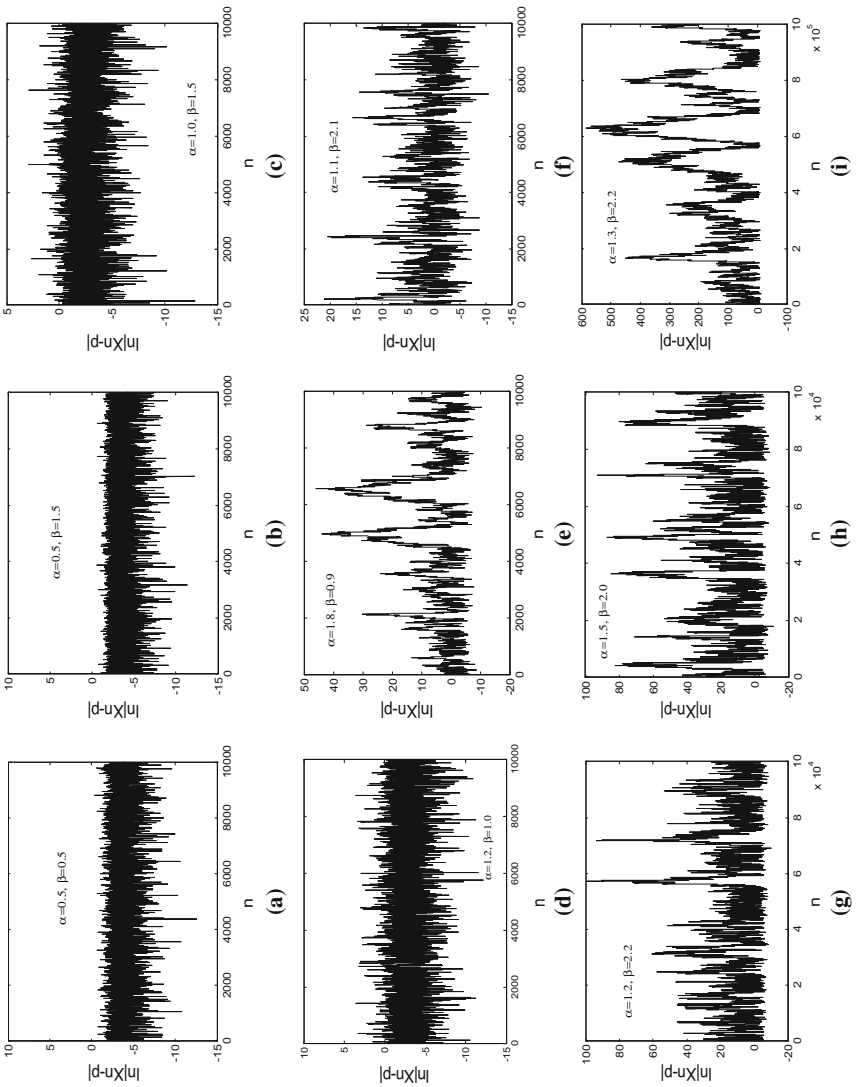

Fig. 1 The figure visualizes the simulation results for the behavior of the particle at different values of α and β. **a**–**c** show that when the values of α and β are selected within the intervals (0, 1) and (0, 2), the particle's position is probabilistically bounded. **d**–**i** show that the particle's position may be also probabilitically bounded at some values of α and β not satisfying constraint (21). **j**–**o** show some cases that when α and β do not satisfy constraint (21), $\ln|X_n - p| \to +\infty$ (i.e.,$|X_n - p| \to +\infty$) as n increases

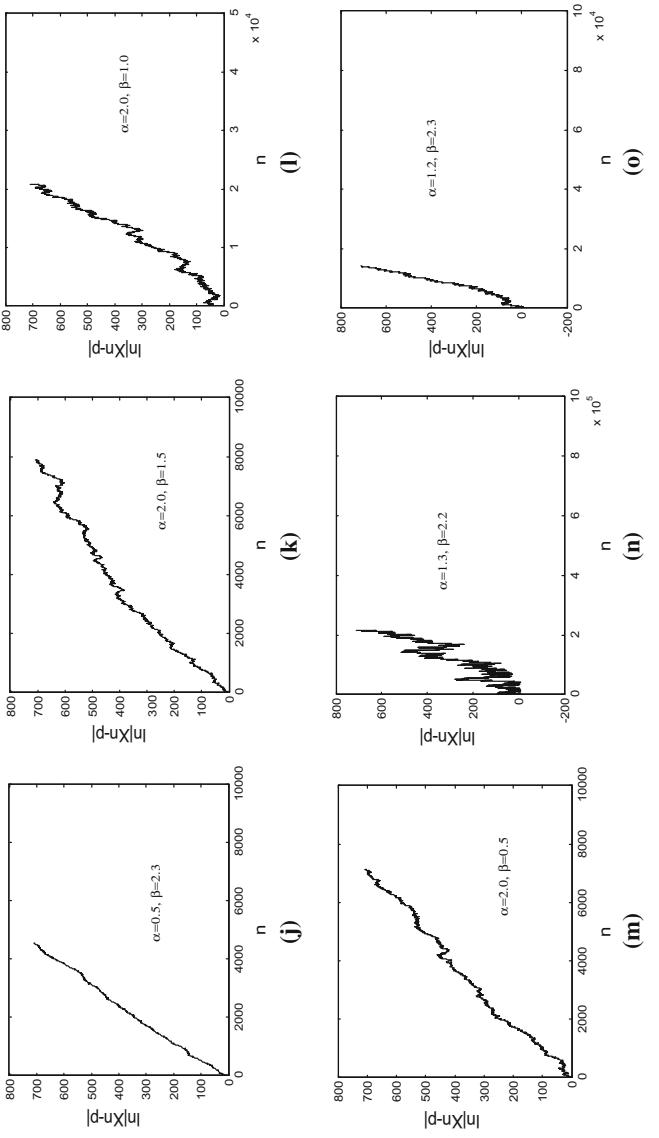

Fig. 1 continued

as independent probabilistically bounded random variables, and thus it is revealed that the behavior of an individual particle is related to the convergence or the boundedness of ρ_n. However, the actual situation is more complex when the RDPSO algorithm runs in a real-world landscape. During the search process, each particle is influenced not only by ρ_n but also by the points C_n and $p_{i,n}$, which cannot be treated as independent random variables anymore but are dependent on the other particles. As for C_n, it is the mean of the *pbest* positions of all the particles, moving with each *pbest* position varying in the course of search. The local focus $p_{i,n}$, is a random point associated with the *pbest* position of particle $i(P_{i,n})$ and the *gbest* position G_n that rotates among the *pbest* positions of the member particles according to their fitness values. In contrast to C_n, $p_{i,n}$, as well as $P_{i,n}$ and G_n, varies more dramatically, since C_n averages the changes of all the *pbest* positions.

Generally, the *pbest* positions of all the particles converge to a single point when the RDPSO algorithm is performing an optimization task, which implies that $P\{\lim_{n\to\infty}|C_n - p_{i,n}| = 0\} = 1$ as indicated in the proof of Theorem 1. Referring to Eqs. (29)–(32), we can infer that if and only if $\Delta < 0$, $\lim_{n\to\infty}|X_{i,n} - C_n| = 0$ or $\lim_{n\to\infty}|X_{i,n} - p_{i,n}| = 0$. That means the current positions and the *pbest* positions of all the particles converge to a single point when $\Delta < 0$. It can also be found from Theorems 2 and 3 that, when $\Delta = 0$, the particle's position is probabilistically bounded and oscillates around but does not converge to C_n or $p_{i,n}$, even though $P\{\lim_{n\to\infty}|C_n - p_{i,n}| = 0\} = 1$. When $\Delta > 0$, it is shown by Theorems 2 and 3 that the particle's current position diverges and the explosion of the whole particle swarm happens.

In practical applications, it is always expected that the particle swarm in the RDPSO algorithm can converge to a single point, just as in the canonical PSO. Essentially, there are two movement trends, i.e. the random motion and the drift motion, for each particle in the RDPSO, as has been described in the motivation of the algorithm. These two motions reflect the global search and the local search, respectively. The drift motion, represented by the $VD^j_{i,n+1}$ in the velocity update Eq. (15), draws the particle towards the local focus and makes the particle search in the vicinity of the *gbest* position and its *pbest* position so that the particle's current and *pbest* positions can constantly come close to the *gbest* position. On the other hand, the random component $VR^j_{i,n+1}$ results in a random motion, leading the particle to be volatile so that its current position may reach a point far from the *gbest* position and its *pbest* position. This component can certainly endue the particle a global search ability, which in the canonical PSO algorithm is given by the velocity at the last iteration, i.e. $V^j_{i,n+1}$. Nevertheless, an important characteristic distinguishing the RDPSO from other randomized PSO methods is that the random component of the particle's velocity uses an adaptive standard deviation for its distribution, i.e. $\alpha|X^j_{i,n} - C^j_n|$. Such a random component makes the random motion of the particle have a certain orientation. The effect of $VR^j_{i,n+1}$ is pulling or pushing the particle away from the *gbest* position by C^j_n as shown by Fig. 2, not just displacing the particle randomly as the mutation operation does in some variants of PSO and evolutionary algorithms. Figure 2a shows that, when C^j_n is at the left side of $X^j_{i,n}$ and G^j_n, $|X^j_{i,n} - C^j_n| = X^j_{i,n} - C^j_n$. The drift component $\beta(p^j_{i,n} - X^j_{i,n})$ draws the particle right towards G^j_n. If $\varphi^j_{i,n+1} > 0$, $\alpha|X^j_{i,n} - C^j_n|\varphi^j_{i,n+1} = \alpha(X^j_{i,n} - C^j_n)\varphi^j_{i,n+1} > 0$, which makes the particle move to the right further and, thus, pushes $X^j_{i,n}$ away from G^j_n. If $\varphi^j_{i,n+1} < 0$, $\alpha(X^j_{i,n} - C^j_n)\varphi^j_{i,n+1} < 0$, whose effect is that the particle's position is pulled away from G^j_n. Figure 2b illustrates the case when C^j_n is at the right side of $X^j_{i,n}$ and G^j_n. Only the effect of the sign of $\varphi^j_{i,n+1}$ on the direction of the particle's motion is opposite to that

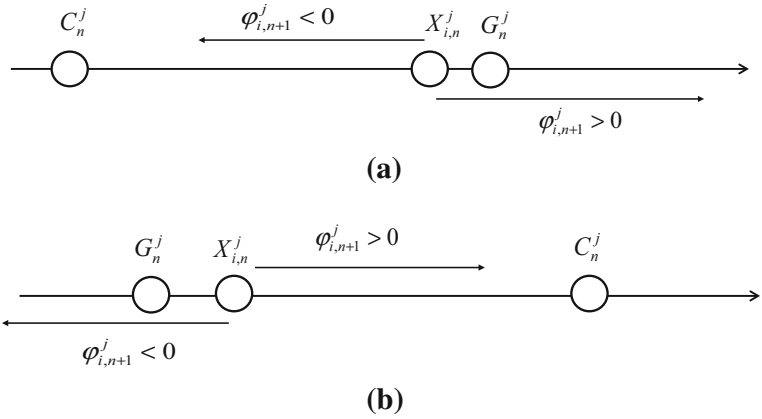

Fig. 2 The figure shows that the *mbest* position C_n^j pulls or pushes the particle away from G_n^j. The direction of the particle's movement is determined by the sign of $\varphi_{i,n+1}^j$

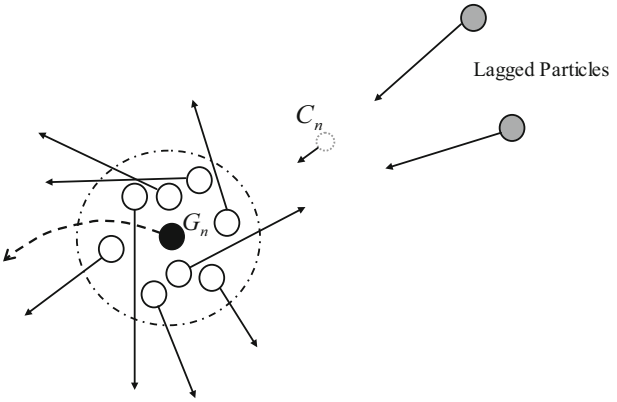

Fig. 3 The figure shows that C_n is shifted toward the lagged particles and thus far from the particles clustering around G_n. The particles are pulled or pushed away from the neighbourhood of G_n and would search the landscape globally

in Fig. 2a. Generally speaking, the longer the distance $|X_{i,n}^j - C_n^j|$, the farther the particle's position at next iteration $X_{i,n+1}^j$ will be away from the *gbest* position. If the particle's position is close to the *gbest* position, the random component can help the particle escape the *gbest* position easily, when the *gbest* position is stuck into a local optimal solution. As far as the whole particle swarm is concerned, the overall effect is that the RDPSO has a better balance between the global search and the local search, as illustrated below.

In the RDPSO method, the swarm could not gather around the *gbest* position without waiting for the lagged particles. Figure 3 depicts the concept where the *pbest* positions of several particles, known as the lagged particles, are located far away from the rest of the particles and the *gbest* position G_n, while the rest of the particles are nearer to the global best position, with their *pbest* positions located within a neighbourhood of the *gbest* position. The *mbest* position C_n would be shifted towards the lagged particles and be located outside the neighbourhood. When the lagged particles chase after their colleagues, that is, converge to

G_n, C_n approaches G_n slowly. The current positions of the particles within the neighbourhood would be pulled or pushed outside the neighbourhood by C_n, and the particles would explore the landscape globally around G_n so that the current G_n could skip out onto a better solution. As C_n careens toward the neighbourhood, the exploration scope of the particle becomes narrower. After the lagged particles move into the neighbourhood of the *gbest* position, C_n also enters the neighbourhood and the particles then perform the same search process based on a smaller neighbourhood of the *gbest* position. In the canonical PSO, each particle converges to the *gbest* position independently and has less opportunity to escape from the neighbourhood of the *gbest* position. When the speed of the particle is small, it is impossible for the particles within the neighbourhood to jump out of the neighbourhood. As a result, these particles would perform local search around the *gbest* position and only the lagged particles could search globally. Evident from the above analysis, the RDPSO algorithm generally has a better balance between exploration and exploitation than the canonical PSO.

Moreover, different from mutation operations that play minor roles in some variants of PSO and evolutionary algorithms, the random motion has an equally important role as the drift motion in the RDPSO. Owing to the random motion oriented by C_n, the RDPSO achieves a good balance between the local and global searches during the search process. By the influences of both C_n and its local focus, each particle in the RDPSO have two movement trends, convergence and divergence, but the overall effect is their convergence to a common point of all the particles if $\Delta < 0$. The convergence rate of the algorithm depends on the values of α and β, which can be tuned to balance the local and global search, when the algorithm is used for a practical problem.

4 Four variants of RDPSO

In order to investigate the RDPSO in depth, some variants of the algorithm are proposed in this paper. Two methods are used for determining the random component of the velocity. One employs Eq. (13) for this component and the other replaces the *mbest* position in Eq. (13) by the*pbest* position of a randomly selected particle in the population at each iteration. For convenience, we denote the randomly selected *pbest* position by C'_n. For each particle, the probability for its *pbest* position to be selected as C'_n is $1/M$. Consequently, the expected value of C'_n equals to C_n, that is,

$$E(C'_n) = \sum_{i=1}^{M} \frac{1}{M} P_{i,n} = C_n. \tag{22}$$

However, as the C'_n appears to be more changeful than C_n, the current position of each particle at each iteration shows to be more volatile than that of the particle with Eq. (13), which diversifies the particle swarm and in turn enhances the global search ability of the algorithm.

In addition to the global best model, the local best model is also examined for the RDPSO, in order to make a comprehensive empirical analysis of the RDPSO algorithm in different neighborhood topologies. The ring topology is a widely used neighborhood topology for the local best model (Li 2010; Engelbrecht 2013), in which each particle connects exactly to two neighbors. The standard PSO (SPSO) in Bratton and Kennedy (2007) is defined by the integration of the PSO-Co with the ring topology. Although there are various neighborhood topologies, we chose the ring topology for the RDPSO with the local best model. Thus, the

combination of the two topologies with the two strategies for the random velocity component produces the four resulting RDPSO variations:

RDPSO-Gbest: The RDPSO algorithm with the global best model and the random velocity component described by Eq. (13).

RDPSO-Gbest-RP: The RDPSO algorithm using the global best model and employing a randomly selected *pbest* position to determine the random velocity component.

RDPSO-Lbest: The RDPSO algorithm with the ring neighborhood topology and the random velocity component in Eq. (13), where, however, the *mbest* position is the mean of the *pbest* positions of the neighbors of each particle and the particle itself, instead of the mean of the *pbest* positions of all the particles in the swarm.

RDPSO-Lbest-RP: The RDPSO algorithm using the ring neighborhood topology and employing the *pbest* position of a particle randomly selected from the neighbors of each particle and the particle itself.

5 Experimental results and discussion

5.1 Benchmark problems

The previous analysis of the RDPSO provides us with a deep insight into the mechanism of the algorithm. However, it is not sufficient to evaluate the effectiveness of the algorithm without comparing it with other methods on a set of benchmark problems. To evaluate the RDPSO in an empirical manner, the first fourteen functions from the CEC2005 benchmark suite (Suganthan et al. 2005) were employed for this purpose. Functions F_1 to F_5 are unimodal, functions F_6 to F_{12} are multi-modal, and F_{13} and F_{14} are two expanded functions. The mathematical expressions and properties of the functions are described in detail in (Suganthan et al. 2005). The codes in Matlab, C and Java for the functions can be found at http://www.ntu.edu.sg/home/EPNSugan/. The dimension of each tested benchmark function in our experiments is 30.

5.2 Empirical studies on the parameter selection of the RDPSO variants

Parameter selection is the major concern when a stochastic optimization algorithm is employed to solve a given problem. For the RDPSO, the algorithmic parameters include the swarm size, the maximum number of iterations, the thermal coefficient α and the drift coefficient β. As in the canonical PSO, the swarm size in the RDPSO is recommended to be set from 20 to 100. The selection of the maximum number of iterations depends on the problem to be solved. In the canonical PSO, the acceleration coefficients and the inertia weight (or the constriction factor) have been studied extensively and in depth since these parameters are very important for the convergence of the algorithm. For the RDPSO algorithm, α and β play the same roles as the inertia weight and the acceleration coefficients for the canonical PSO. In Sect. 3, it was shown that it is sufficient to set α and β according to (21), such that $\Delta < 0$, to prevent the individual particle from divergence and guarantee the convergence of the particle swarm. However, this does not mean that such values of α and β can lead to a satisfactory performance of the RDPSO algorithm in practical applications. This section intends to find out, through empirical studies, suitable settings of α and β so that the RDPSO may yield satisfactory algorithmic performance in general.

There are various control methods for the parameters α and β when the RDPSO is applied to practical problems. A simple approach is to set them as fixed values when the algorithm

is executed. Another method is to decrease the value of the parameter linearly during the course of the search process. In this work, we fixed the value of β in all the experiments and employed the two control methods for α, respectively.

To specify the value of α and β for real applications of the RDPSO, we tested the RDPSO-Gbest, RDPSO-Gbest-RP, RDPSO-Lbest, and RDPSO-Lbest-RP with different parameter settings on three frequently used functions from the CEC2005 benchmark suite: Shifted Rosenbrock Function (F_6), Shifted Rotated Griewank's Function (F_7), and Shifted Rastrigin's Function (F_9), using the two methods for controlling α with β fixed at 1.5 or 1.45. The initial position of each particle was determined randomly within the initialization range. One reason why only three functions were used for parameter selection is that we want to show that the RDPSO algorithm is not very sensitive to the parameter values, and parameter values found by optimizing these three functions can lead to good performance when optimizing other functions in general. Another reason is that these three functions are widely used in the existing literature and that the optimal parameter values for each function are very different, that is, the optimal parameter values for a function may have a poor performance when used for another function.

For each parameter configuration, each algorithm, using 40 particles, was tested for 100 runs on every benchmark function. To determine the effectiveness of each algorithm for the α setting under a control method with a fixed value of β on each problem, the best objective function value (i.e., the best fitness value) found after 5000 iterations was averaged over 100 runs of tests for that parameter setting and the same benchmark function. The results (i.e., the mean best fitness values) obtained by the parameter settings with the same control method for α were compared across the three benchmarks. The best parameter setting with each control method for α was selected by ranking the averaged best objective function values for each problem, summing the ranks, and taking the value that had the lowest summed (or average) rank, provided that the performance is acceptable (in the top half of the rankings) in all the tests for a particular parameter configuration.

The rankings of the results for the RDPSO-Gbest are plotted in Fig. 4. When the fixed value method was used for α, it was set to a range of values subject to constraint (21), with β fixed at 1.5 or 1.45 in each case. Results obtained for other parameter settings were very poor and are not considered for ranking. The best average rank among all the tested parameter configurations occurs when $\alpha = 0.7$ and $\beta = 1.5$. When linearly varying α was used, its initial value α_1 and final value α_2 ($\alpha_1 > \alpha_2$) were selected from a series of different values subject to constraint (21), with β set at 1.5 or 1.45. Only acceptable results are ranked and plotted in Fig. 4. It was found that with $\beta = 1.45$, decreasing α linearly from 0.9 to 0.3 leads to the best performance among all the tested parameter settings.

The rankings of the results for the RDPSO-Gbest-RP are visualized in Fig. 5. It is clear from these results that the value of α, whether it used the fixed value or time-varying method, should be set relatively small, so that the algorithm is comparable in performance with the RDPSO-Gbest, when β was given. Results obtained with α outside the range [0.38, 0.58] were of poor quality and were not used for ranking. As shown in Fig. 5, when the fixed value method for α was used, the best average ranks among all tested parameter settings were obtained by setting $\alpha = 0.5$ and $\beta = 1.45$. On the other hand, the algorithm exhibited the average best performance when $\beta = 1.45$ and α was decreasing linearly from 0.6 to 0.2, for the method of linearly varying α.

Figure 6 shows the rankings of the results for the RDPSO-Lbest. For the fixed α method, the results of the algorithm obtained with α outside the range [0.6, 0.78] did not participate in ranking because of their poor qualities. The best average ranking among all the tested parameter configurations in this case occur when $\alpha = 0.7$ and $\beta = 1.5$. For the linearly

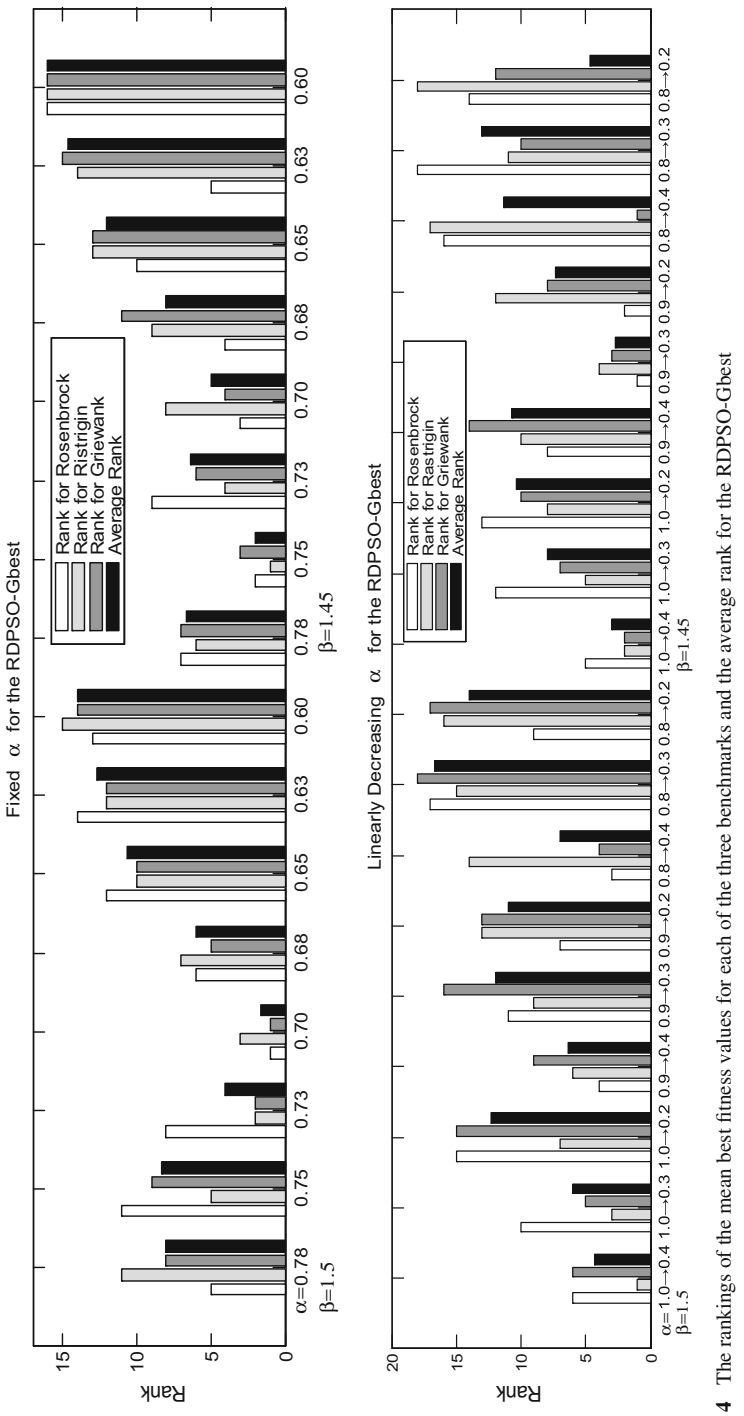

Fig. 4 The rankings of the mean best fitness values for each of the three benchmarks and the average rank for the RDPSO-Gbest

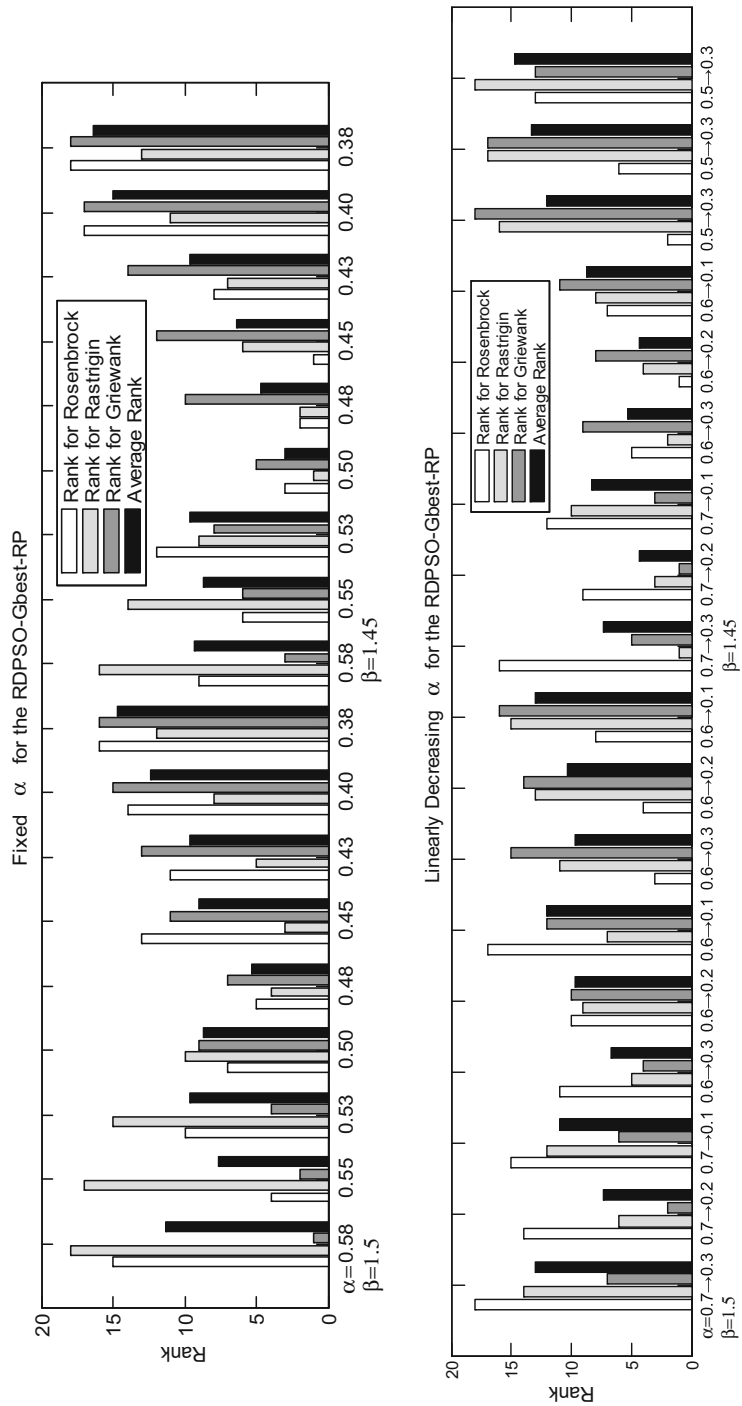

Fig. 5 The rankings of the mean best fitness values for each of the three benchmarks and the average rank for the RDPSO-Gbest-RP

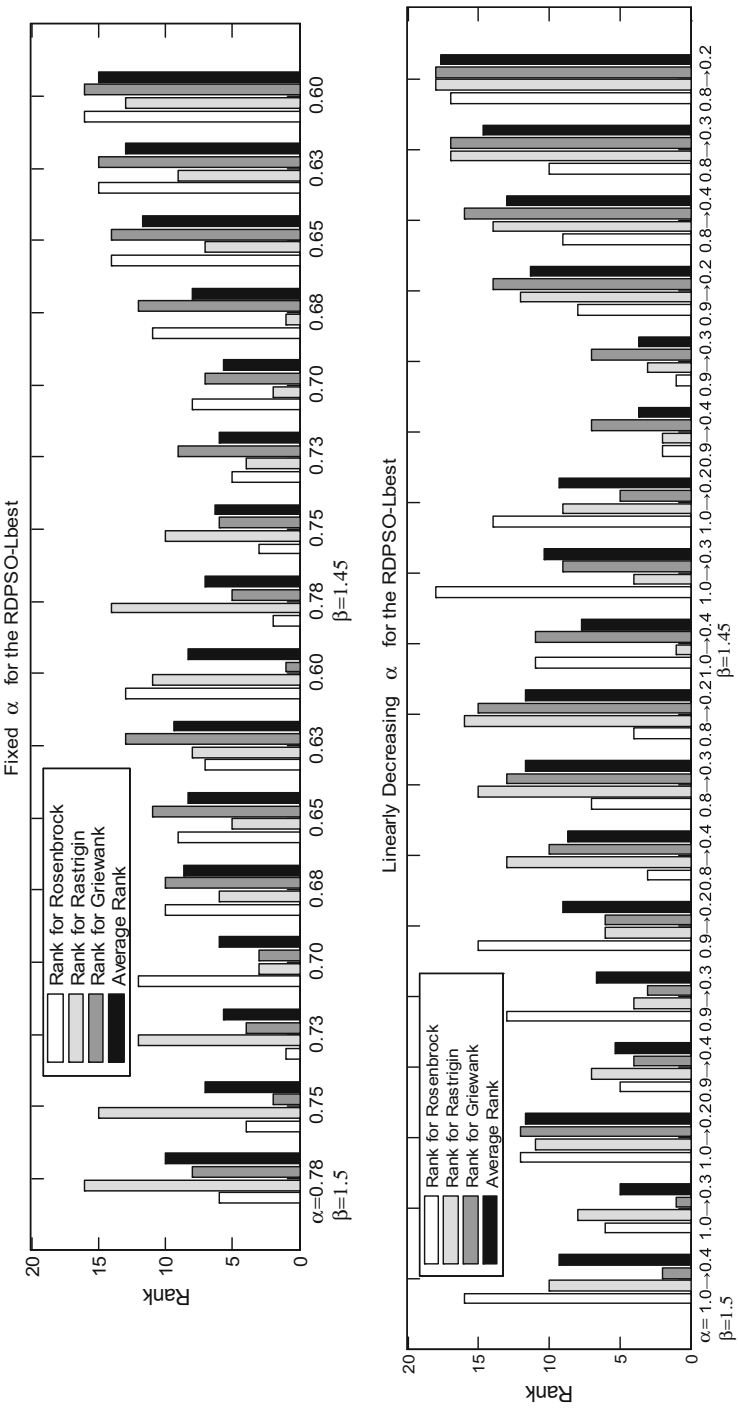

Fig. 6 The rankings of the mean best fitness values for each of the three Benchmarks and the average rank for the RDPSO-Lbest

varying α method, it was identified that decreasing α linearly from 0.9 to 0.3 with $\beta = 1.45$ could yield the average best quality results among all the tested parameter configurations.

Figure 7 plots the rankings of the results for the RDPSO-Lbest-RP. For fixed α, the best average ranking among all the tested parameter settings could be obtained when $\alpha = 0.7$ and $\beta = 1.45$. For time-varying α, the algorithm obtained the average best performance among all the tested parameter configurations when α was decreasing linearly from 0.9 to 0.3 with $\beta = 1.45$.

5.3 Performance comparisons among the RDPSO variants and other PSO variants

To explore the generalizability of the parameter selection methods for α used for the RDPSO in the last subsection, and to the determine whether RDPSO can be as effective as other variants of PSO, a further performance comparison using the first fourteen benchmark functions of the CEC2005 benchmark suite was made among the RDPSO algorithms (i.e., the RDPSO-Gbest, RDPSO-Gbest-RP, RDPSO-Lbest and RDPSO-Lbest-RP) and other PSO variants, including the PSO with inertia weight (PSO-In) (Shi and Eberhart 1998a, b, 1999), the PSO with constriction factor (PSO-Co) (Clerc and Kennedy 2002; Clerc 1999), the PSO-In with local best model (PSO-In-Lbest) (Liang et al. 2006), the standard PSO (SPSO) (i.e. PSO-Co-Lbest) (Bratton and Kennedy 2007), the Gaussian bare bones PSO (GBBPSO) (Kennedy 2003, 2004), the comprehensive learning PSO (CLPSO) (Liang et al. 2006), the dynamic multiple swarm PSO (DMS-PSO) (Liang and Suganthan 2005), and the fully-informed particle swarm (FIPS) (Mendes et al. 2004). Each algorithm was run 100 times for each benchmark function, using 40 particles to search the global optimal fitness value. At each run, the particles in the algorithms started in new and randomly-generated positions, which are uniformly distributed within the search bounds. Each run of every algorithm lasted for 5000 iterations, and the best fitness value (objective function value) for each run was recorded.

For the four RDPSO variants, it was shown in the last subsection that the linearly decreasing α with fixed β was stable in the search performance, although fixing both α and β had better results in some cases. Thus, in this group of experiments for performance comparison, the linearly decreasing α with fixed β was used for the RDPSO variants, and the parameters for each case were set as those indentified and recommended by the previous experiments on the three benchmark functions. These parameter configurations were selected from the experiments on the three functions, so they are far from optimal. The parameter configurations for other PSO variants were the same as those recommended by the existing publications. For the PSO-In, the inertia weight linearly decreased from 0.9 to 0.4 in the course of the run and we fixed the acceleration coefficients (c_1 and c_2) at 2.0, as in the empirical study performed by Shi and Eberhart (1999). For the PSO-Co, the constriction factor was set to be $\chi = 0.7298$, and the acceleration coefficients $c_1 = c_2 = 2.05$, as recommended by Clerc and Kennedy (2002). Eberhart and Shi also used these values of the parameters when comparing the performance of the PSO-Co with that of the PSO-In (Eberhart and Shi 2000). For the SPSO, the ring topology was used and other parameters were set as those in the PSO-Co (Bratton and Kennedy 2007). Parameter configurations for the GBBPSO, FIPS, DMS-PSO and CLPSO were the same as those in Kennedy (2003), Mendes et al. (2004), Liang and Suganthan (2005), Liang et al. (2006), respectively. The justification for using the recommended parameter settings for these PSO variants is that in their related papers, the parameter configurations for these algorithms were tested on different benchmark functions, including those three functions used in our experiments for the RDPSO. The performance of these parameter settings were satisfactory and, thus, were recommended by the authors.

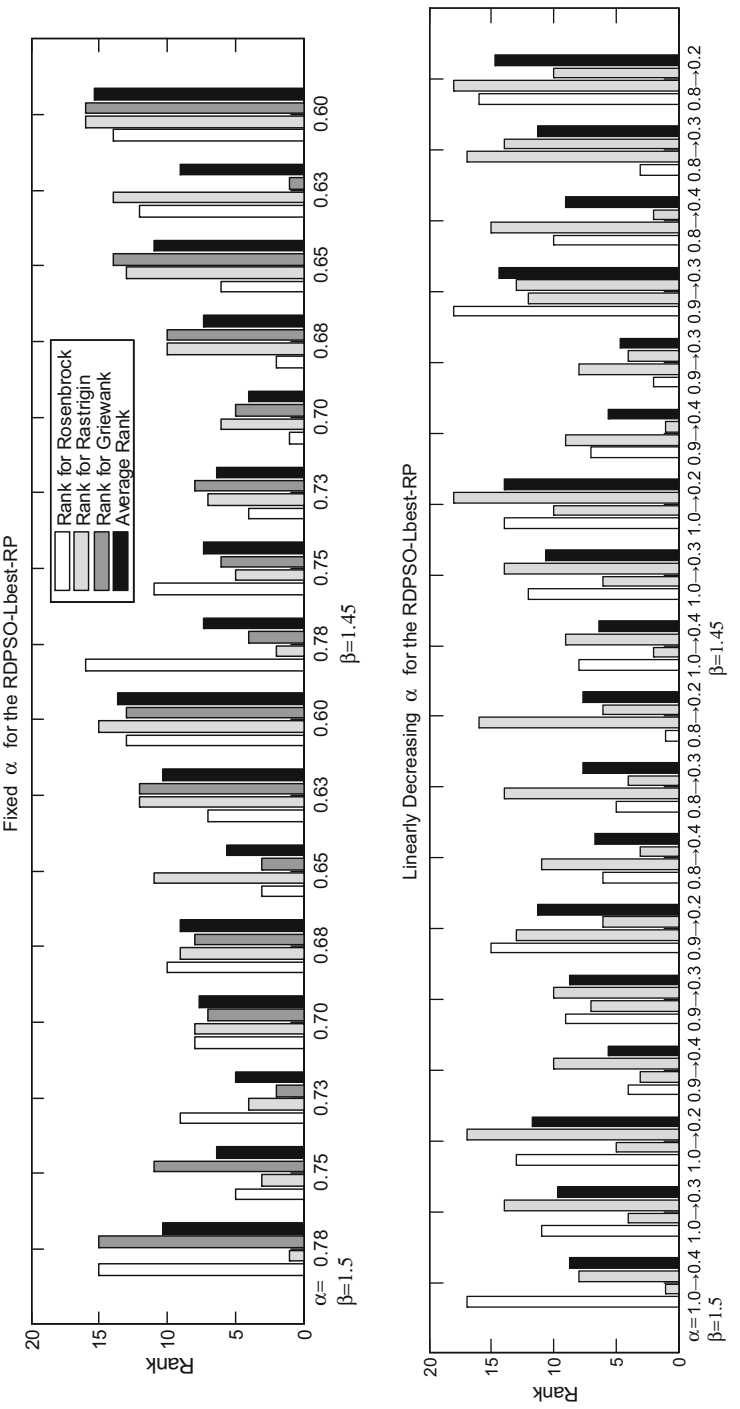

Fig. 7 The rankings of the mean best fitness values for each of the three Benchmarks and the average rank for the RDPSO-Lbest-RP

Tables 1 and 2 record the mean and the standard deviation of the best fitness values out of 100 runs of each algorithm on each benchmark function. The bold in the tables corresponds to the smallest ones among the mean best fitness values and the standard deviations obtained for each function by all the compared algorithms. To investigate whether the differences in the mean best fitness values among the algorithms were significant, a statistical multiple comparison procedure was implemented to determine the algorithmic performance ranking for each problem in a statistical manner. The procedure employed in this work is known as the "stepdown" procedure (Day and Quinn 1989). The algorithms that were not statistically different to each other were given the same rank; those that were not statistically different to more than one other groups of algorithms were ranked with the best-performing of these groups. For each algorithm, the resulting rank for each problem and the average rank across all the tested fourteen benchmark problems are shown in Table 3.

For the Shifted Sphere Function (F_1), the RDPSO-Lbest-RP generated better results than the other methods. The results for the Shifted Schwefel's Problem 1.2 (F_2) show that the PSO-Co and the GBBPSO performed better than the others, but the performance of the CLPSO seems to be inferior to those of other competitors due to its slow convergence speed. For the Shifted Rotated High Conditioned Elliptic Function (F_3), the RDPSO-Gbest-RP outperformed the other methods in a statistical significance manner. The SPSO was the second best performing method for this function. The RDPSO-Gbest-RP showed to be the winner among all the tested algorithms for the Shifted Schwefel's Problem 1.2 with Noise in Fitness (F_4), and the RDPSO-Gbest was the second best performing for this problem. F_5 is the Schwefel's Problem 2.6 with Global Optimum on the Bounds. For this benchmark, the RDPSO-Gbest-RP occupied the first place from the perspective of the statistical test. For benchmark F_6, the Shifted Rosenbrock Function, both the RDPSOs with the ring topology outperformed the other algorithms. The results for the Shifted Rotated Griewank's Function without Bounds (F_7) suggest that both the RDPSOs with local best model and the SPSO were able to find the solution to the function with better quality compared to the other methods. Benchmark F_8 is the Shifted Rotated Ackley's Function with Global Optimum on the Bounds. The SPSO and the PSO-In-Lbest yielded better results for this problem than the others. The Shifted Rastrigin's Function (F_9) is a separable function, which the CLPSO algorithm was good at solving and obtained remarkably better results for. It can also be observed that the RPDOS-Gbest yielded a better result than the remainders. F_{10} is the Shifted Rotated Rastrigrin's Function, which appears to be a more difficult problem than F_9. For this benchmark, both the RDPSO-Lbest and RDPSO-Lbest-RP outperformed the other competitors in a statistically significant manner. The best result for the Shifted Rotated Weierstrass Function (F_{11}) was obtained by the RDPSO-Gbest-RP. The RDPSO-Gbest yielded the second best result which shows no statistical significance with that of the RDPSO-Gbest-RP. When searching the optima of Schewefel's Problem 2.13 (F_{12}), the RDPSO-Gbest-RP was found to rank first in algorithmic performance from a statistical point of view.

F_{13} is the Shifted Expand Griewank's plus Rosenbrock's Function, for which the RDPSO-Lbest-RP, RDPSO-Lbest, and RDPSO-Gbest yielded better results than their competitors. There are no statistically significant differences in algorithmic performance between the three RDPSO variants. For the Shifted Rotated Expanded Scaffer's $F6$ Function (F_{14}), all the RDPSO variants showed better performance than the others in a statistically significant manner.

The average ranks listed in Table 3 reveal that the RDPSO-Gbest-RP had the best overall performance for the fourteen benchmark functions among all the tested algorithms. Across the whole suite of benchmark functions, it had fairly stable performance with the worst

Table 1 Mean and standard deviation of the best fitness values after 100 runs of each algorithm for F_1 to F_7

Algorithms	F_1	F_2	F_3	F_4	F_5	F_6	F_7
PSO-In	3.9971e−028	263.2219	3.4324e+007	2.7829e+003	4.3961e+003	143.7144	0.3285
	(5.6544e−028)	(608.4657)	(3.0220e+007)	(2.0996e+003)	(1.5331e+003)	(336.9297)	(1.1587)
PSO-Co	6.7053e−029	**0.0100**	1.3659e+007	842.4768	6.2857e+003	57.5740	0.0283
	(1.0671e−028)	(**0.0939**)	(1.3662e+007)	(1.5264e+003)	1.9629e+003	(84.2278)	(0.0184)
PSO-In-Lbest	2.7049e−013	865.7861	2.5658e+007	8.7648e+003	8.0095e+003	57.5362	0.1830
	(5.1148e−013)	(368.9980)	(1.0089e+007)	(1.8468e+003)	(1.0568e+003)	(74.5821)	(0.1093)
SPSO	4.2657e−036	0.8615	3.3604e+006	6.3348e+003	5.2549e+003	47.3744	0.0108
(PSO-Co-Lbest)	(2.3958e−036)	(0.7092)	(1.5549e+006)	(2.3147e+003)	(1.1583e+003)	(79.8406)	(0.0078)
GBBPSO	7.0941e−027	0.0110	4.9003e+006	1.0432e+003	8.0391e+003	109.8415	0.0179
	(1.9421e−026)	(0.0174)	(2.6581e+006)	1.0819e+003	(2.8824e+003)	(330.4848)	(0.0170)
FIPS	1.2395e−036	0.1390	6.9970e+006	4.5429e+003	3.3929e+003	109.1170	0.0147
	(8.4958e−037)	(0.0682)	(2.4490e+006)	(1.4685e+003)	(599.5893)	(179.8489)	(0.0101)
DMS-PSO	8.8399e−016	141.1109	5.6008e+006	976.6745	2.4263e+003	211.0941	0.0283
	(2.1311e−015)	(70.6632)	(2.9187e+006)	(391.0695)	(498.7101)	(314.9179)	(0.0226)
CLPSO	5.2323e−017	1.2661e+003	3.3326e+007	7.6045e+003	4.0357e+003	74.2914	1.0054
	(2.9219e−017)	(297.3666)	(8.8808e+006)	(1.7722e+003)	(489.0741)	(31.5737)	(0.0663)
RDPSO-Gbest	2.2871e−027	0.0805	4.7079e+006	411.2758	2.6293e+003	60.9164	0.0175
	(4.3476e−028)	(0.1341)	(3.1653e+006)	(574.1945)	(808.8539)	(78.5198)	(0.0140)
RDPSO-Gbest-RP	8.1256e−037	0.1131	**2.5203e+006**	**217.8821**	**2.2241e+003**	34.9274	0.0130
	(1.4983e−037)	(1.0156)	(**1.6334e+006**)	(**269.0046**)	(**865.3596**)	(39.0403)	(0.0123)
RDPSO-Lbest	3.9443e−031	2.4034	4.9772e+006	1.6199e+003	2.7654e+003	**19.5009**	**0.0092**
	(9.5470e−031)	(1.7191)	(1.9029e+006)	(883.4518)	(638.3375)	(**16.7704**)	(**0.0050**)
RDPSO-Lbest-RP	**5.2461e−037**	9.3880	4.8092e+006	3.4502e+003	3.9088e+003	24.0065	0.0093
	(**7.3587e−038**)	(6.7340)	(1.7477e+006)	(1.3764e+003)	(888.7718)	(24.4861)	(0.0061)

Table 2 Mean and standard deviation of the best fitness values after 100 runs of each algorithm for F_8 to F_{14}

Algorithms	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
PSO-In	21.1149	28.1848	214.2491	38.6029	3.0743e+004	5.2896	13.8002
	(0.0650)	(11.4742)	(84.8990)	(7.9234)	(2.9043e+004)	(5.5476)	(0.3444)
PSO-Co	21.1271	71.0598	123.1232	26.6597	1.0415e+004	4.4108	12.7952
	(0.0557)	(22.0534)	(51.0717)	(5.1673)	(1.3897e+004)	(1.2793)	(0.4972)
PSO-In-Lbest	20.9274	39.0149	149.9040	29.4701	1.6420e+004	5.1283	13.0249
	(0.0518)	(8.0007)	(39.4806)	(2.2549)	(8.2755e+003)	(1.3492)	(0.2546)
SPSO	**20.9092**	65.1992	90.4544	29.1374	4.5191e+003	4.1371	12.6110
(PSO-Co-Lbest)	**(0.0592)**	(13.3166)	(18.4968)	(2.1661)	(3.3662e+003)	(0.8434)	(0.2924)
GBBPSO	20.9631	60.3143	127.2546	28.2383	1.7318e+004	4.9260	13.5393
	(0.0481)	(15.3916)	(48.5001)	(3.4455)	(6.4095e+004)	(1.3859)	(0.5470)
FIPS	20.9638	47.9595	170.4301	32.6119	3.1169e+004	8.4372	12.7804
	(0.0476)	(9.9315)	(19.0757)	(2.5941)	(1.5581e+004)	(1.3535)	(0.2627)
DMS-PSO	20.9569	29.5427	77.6689	23.8535	7.4986e+003	5.1709	12.6673
	(0.0522)	(7.4630)	(11.9670)	(2.1849)	(6.2259e+003)	(1.7631)	(0.3139)
CLPSO	20.9613	**7.3197e−006**	118.2419	23.8084	3.4442e+004	3.8576	13.1524
	(0.0499)	**(1.2443e−005)**	(14.6277)	(2.1761)	(7.6392e+003)	(0.3906)	(0.1691)
RDPSO-Gbest	20.9558	22.7650	78.6024	21.6689	6.0361e+003	3.6656	12.4291
	(0.0641)	(5.7728)	(38.9282)	(7.6173)	(5.1260e+003)	(1.6923)	(0.4285)
RDPSO-Gbest-RP	20.9602	31.9085	82.5152	**20.0701**	**2.8227e+003**	4.1837	12.5091
	(0.0569)	(8.7969)	(47.7362)	**(6.9879)**	**(3.3963e+003)**	(2.9678)	(0.4033)
RDPSO-Lbest	20.9540	27.8237	**49.8606**	22.1984	4.0616e+003	3.5717	12.4730
	(0.0508)	(6.0386)	**(12.9486)**	(3.0396)	(2.8103e+003)	(1.1045)	(0.2576)
RDPSO-Lbest-RP	20.9613	36.4589	51.4390	23.0731	3.7315e+003	**3.3328**	**12.3497**
	(0.0543)	(7.9391)	(7.9391)	(1.8929)	(1.9675e+003)	**(0.7410)**	**(0.7410)**

Table 3 Ranking by algorithms and problems obtained from "Stepdown" multiple comparisons

Algorithms	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	*Ave. rank*	*Final rank*
PSO-In	7	=9	=11	7	8	=9	11	15	=3	12	12	=10	=8	12	9.57	12
PSO-Co	6	=1	9	5	10	7	=8	15	12	=7	7	7	=5	=7	7.57	9
PSO-In-Lbest	12	11	10	12	=11	=3	10	=1	=7	10	10	=8	=8	9	8.71	11
SPSO	4	6	2	10	9	=3	=1	=1	=10	=3	=8	4	=5	=5	5.07	5
GBBPSO	9	=1	=3	=3	=11	=9	=4	=3	=10	=7	=8	=8	=8	11	6.79	7
FIPS	3	=3	8	9	5	=9	=4	=3	9	11	11	=10	12	=7	7.43	8
DMS-PSO	11	=9	=3	=3	=1	12	=8	=3	=3	=3	=4	=5	=8	=5	5.57	6
CLPSO	10	12	=11	11	=6	=8	12	=3	1	=7	=4	=10	4	10	7.79	10
RDPSO-Gbest	8	=3	=3	2	=1	=3	=4	=3	2	=3	=1	=5	=1	=1	2.86	3
RDPSO-Gbest-RP	2	=3	1	1	=1	=3	=4	=3	6	=3	=1	=1	=5	=1	2.50	1
RDPSO-Lbest	5	7	=3	6	4	=1	=1	=3	=3	=1	=1	=1	=1	=1	2.71	2
RDPSO-Lbest-RP	1	8	=3	8	=6	=1	=1	=3	=7	=1	=4	=1	=1	=1	3.29	4

rank being 6 for F_9. The second best-performing was the RPDSO-Lbest. For seven of the benchmark functions, the algorithm had the first performance ranks. However, its result for F_2 is unsatisfactory due to its slow convergence speed. The RDPSO-Gbest had the third best overall performance. Compared to the RDPSO-Gbest-RP, the RDPSO-Gbest performed somewhat unstable, with the resulting ranks for F_1 being only 8. The fourth best performing was the RDPSO-Lbest-RP, which did not show satisfactory performance on F_2 and F_4. Nevertheless, it had a significant advantage over the SPSO, the next best performing one. Between random velocity components determined by the *mbest* position and the random selected *pbest* position, the two versions of the RDPSO with the *mbest* position obtained the total average rank of 2.79, while the two with the randomly selected *pbest* position had the total average rank of 2.90. This means that there is no remarkable performance difference for the tested functions between the two different methods for determining random velocity components. What can be found from the total average ranks is that the RDPSO algorithms were able to perform better by using the global best model (with the total average rank of 2.53) than the local best model (with the total average ranks of 3.00) for the first fourteen CEC2005 benchmark functions. In addition, the total average rank over all the versions of the RDPSO is 2.84, which implies that the RDPSOs with the linearly varying α and fixed β had a satisfactory overall performance. Therefore, it is recommended that the linearly varying α method with fixed β should be employed when the RDPSO is used for real applications with the values of the parameters tuned finely around the values used in the experiments in this work. More specifically, for the RDPSO-Gbest, RDPSO-Lbest and RDPSO-Lbest-RP, the initial and final values of α can be selected from the intervals [0.8, 1.0] and [0.2, 0.4], respectively, depending on the problem to be solved. For the RDPSO-Gbest-RP, the initial and final values of α can be selected from the intervals [0.5, 0.7] and [0.1, 0.3], respectively. The drift coefficient β can be valued on the interval [1.45, 1.5] for all he RDPSO variants.

Except the RDPSO algorithms, the best-performing algorithm was the SPSO, i.e. the PSO-Co-Lbdest, which yielded the best results for F_7 and F_8. The next best algorithm was DMS-PSO, obtaining the first performance rank for F_3 and the worst rank for F_6. The GBBPSO was the next best-performing method. This is an important probabilistic PSO variant and had good performance for unimodal functions. The FIPS, which also employs the ring topology, ranked the first when optimizing F_2. From the total average ranks in Table 3, it is conclusive that incorporating the ring topology into the PSO-In and the PSO-Co could enhance the overall performance of the two PSO variants on the tested benchmark functions. What should be noticed is that the CLPSO is very effective in solving separable functions such as F_9, but not in the rotated functions and unimodal ones due to its slower convergence speed, as has been indicated in the related publication (Liang et al. 2006).

6 Conclusion

In this paper, based on our preliminary previous work, we made a comprehensive study on the RDPSO algorithm, by analyzing the particle behavior and the search mechanism of the algorithm and empirically investigating the four newly proposed variants of the RDPSO algorithm.

A comprehensive analysis of the RDPSO algorithm and its variants was made in order to have a better understanding of the mechanism behind the algorithm. Firstly, the stochastic dynamical behavior of a single particle in the RDPSO was analyzed theoretically. We derived the sufficient and necessary condition as well as a sufficient condition for the particle's

current position to be probabilistically bounded. Secondly, the search behavior of the RDPSO algorithm itself was investigated by analyzing the interaction between the particles, and it was found that the RDPSO may have a good balance between the global and the local search, due to the designed random component of the particle's velocity. In addition, four variants of the RDPSO algorithm were proposed by combining different random velocity components with different neighborhood topologies.

Empirical studies on the RDPSO algorithm were carried out on the first fourteen benchmark functions of the well-known CEC2005 benchmark suite. Two methods of controlling the algorithmic parameters were employed, and each RDPSO variant, with each control method, was first tested on three of the benchmark functions in order to identify the parameter values that can generate satisfactory algorithmic performance. Then, the RDPSO variants with linearly decreasing thermal coefficients and fixed drift coefficients, which were identified to have stable algorithmic performance, were further compared with other forms of PSO on the fourteen functions. The experimental results show that the RDPSO algorithm is comparable with, or even better, than the other compared PSO variants in finding the optimal solutions of the tested benchmark functions.

Acknowledgments This work is supported by Natural Science Foundation of Jiangsu Province, China (Project Number: BK2013161), by National Natural Science Foundation of China (Project Numbers 61170119, 61105128, 61373055), by the Program for New Century Excellent Talents in University (Project Number: NCET-11-0660), by the RS-NSFC International Exchange Programme (Project Number: 61311130141), and by Key grant Project of Chinese Ministry of Education (Project Number: 311024).

Appendix

Theorem 1 *If there is only drift motion for the particle, i.e.* $V^j_{i,n+1} = VD^j_{i,n+1}$ *a sufficient condition for* $X^j_{i,n+1}$ *to converge to* $p^j_{i,n}$ *is* $0 < \beta < 2$.

Proof From Eqs. (14) and (16), we can find that

$$X^j_{i,n+1} - p^j_{i,n+1} = (1-\beta)\left(X^j_{i,n} - p^j_{i,n}\right) + p^j_{i,n} - p^j_{i,n+1}, \tag{23}$$

When the RDPSO algorithm is running, the personal best positions of all the particles converge to the same point. Consequently, $\{p^j_{i,n}\}$ is a convergent Cauchy sequence such that $\lim_{n\to\infty} |p^j_{i,n} - p^j_{i,n+1}| = 0$. Since $0 < \beta < 2$, $1 - |1-\beta| > 0$. Thus, it holds that

$$\lim_{n\to\infty}\left[\left|p^j_{i,n} - p^j_{i,n+1}\right| / (1 - |1-\beta|)\right] = 0,$$

which means that for any $\varepsilon > 0$, there exists an integer $K > 0$ such that whenever $n \geq K$,

$$\left|p^j_{i,n} - p^j_{i,n+1}\right| < \varepsilon \cdot (1 - |1-\beta|). \tag{24}$$

Therefore, from inequality (24), we have

$$\left|X^j_{i,n+1} - p^j_{i,n+1}\right| - \varepsilon \leq |1-\beta|\left(\left|X^j_{i,n} - p^j_{i,n}\right| - \varepsilon\right). \tag{25}$$

This implies that for any $n > K$,

$$\left|X^j_{i,n} - p^j_{i,n}\right| \leq \varepsilon + \left(\prod_{k=K}^{n-1} |1-\beta|\right)\left|X^j_{i,K} - p^j_{i,K}\right|. \tag{26}$$

Since $0 \leq |1-\beta| < 1$, $\lim_{n\to\infty} \prod_{k=K}^{n-1} |1-\beta| = \lim_{n\to\infty} |1-\beta|^{n-K} = 0$. Hence

$$\lim_{n\to\infty} \sup \left| X_{i,n}^{j} - p_{i,n}^{j} \right| \leq \varepsilon. \tag{27}$$

As ε is arbitrary and $|X_{i,n}^{j} - p_{i,n}^{j}| \geq 0$, therefore

$$\lim_{n\to\infty} \left| X_{i,n}^{j} - p_{i,n}^{j} \right| = 0. \tag{28}$$

This completes the proof of the theorem. □

Theorem 2 *The necessary and sufficient condition for the position sequence of the particle* $\{X_n\}$ *to be probabilistically bounded is that* $\rho_n = \prod_{i=1}^{n} \lambda_i$ *does not diverge, namely,* ρ_n *is probabilistically bounded (i.e.* $P\left\{\sup_{n\geq 1} \rho_n < \infty\right\} = 1$.

Proof From Eqs. (17) and (18), the update equation of the particle's position is given by

$$X_{n+1} = \alpha(X_n - C)\varphi_{n+1} - \beta(X_n - p) + X_n, \tag{29}$$

from which we immediately have

$$\begin{aligned} X_{n+1} - C &= \alpha(X_n - C)\varphi_{n+1} - \beta(X_n - p) + X_n - C \\ &= [\alpha\varphi_{n+1} + (1-\beta)](X_n - C) + \beta(p - C) \\ &= \lambda_{n+1}(X_n - C) + \beta(p - C). \end{aligned} \tag{30}$$

Since λ_{n+1} is a continuous random variable, $P\{\lambda_{n+1} = 1\} = 0$. Considering that $\beta(p-C)$ is probabilistically bounded, we have that $r = \frac{\beta(p-C)}{1-\lambda_{n+1}}$ is also a probabilistically bounded random variable. From (30), we can obtain

$$X_{n+1} - C - r = \lambda_{n+1}(X_n - C - r), \tag{31}$$

From which we can recursively derive the following formula

$$X_n = (X_0 - C - r)\prod_{i=1}^{n} \lambda_i + C + r. \tag{32}$$

Since $X_0 - C - r$ is probabilistically bounded, X_n is probabilistically bounded if and only if $\rho_n = \prod_{i=1}^{n} \lambda_i$ is probabilistically bounded. This completes the proof of the theorem. □

Theorem 3 *Let* $\Delta = E(\xi_n) = \frac{1}{\sqrt{2\pi}\alpha} \int_{-\infty}^{+\infty} \ln|x| e^{-\frac{[x-(1-\beta)]^2}{2\alpha^2}} dx$, *where* $\xi_n = \ln|\lambda_n|$ *and* $\lambda_n \sim N(1-\beta, \alpha^2)$. *(1) The necessary and sufficient condition for* $\rho_n = \prod_{i=1}^{n} \lambda_i$ *to converge to zero with probability 1 is* $\Delta < 0$. *(2) The necessary and sufficient condition for* ρ_n *to be probabilistically bounded, i.e.* $P\{\sup \rho_n < \infty\} = 1$, *is* $\Delta \leq 0$.

Proof By Kolmogorov's strong law of large numbers (Shiryayev 1984), it holds that

$$P\left\{\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \xi_i = E(\xi_1) = \Delta\right\} = 1, \tag{33}$$

which is equivalent to the proposition that $\forall m \in Z^+, \exists K_1 \in Z^+$ such that whenever $k \geq K_1$,

$$\Delta - \frac{1}{m} < \frac{1}{k} \sum_{i=1}^{k} \ln|\lambda_i| < \Delta + \frac{1}{m}. \tag{34}$$

(1) *Proof of the necessity*. If $P\{\lim_{n\to\infty}\rho_n = 0\} = 1$, we have that $P\{\lim_{n\to\infty}\ln|\rho_n| = \lim_{n\to\infty}\sum_{i=1}^{n}\ln|\lambda_i| = -\infty\} = 1$, that is, $\forall m \in Z^+, \exists K_2 \in Z^+$, such that whenever $k \geq K_2 \quad \sum_{i=1}^{k}\ln|\lambda_i| < -m$ and thus

$$\frac{1}{k}\sum_{i=1}^{k}\ln|\lambda_i| < -\frac{m}{k}. \tag{35}$$

Therefore, $\forall m \in Z^+$, there exists $K = \max(K_1, K_2)$ such that whenever $k \geq K$, both inequalities (34) and (35) holds, from which we have, $\Delta - 1/m < -m/k$, namely, $\Delta < 1/m - m/k$. Let $k \to \infty$, and considering the artibrariness of $1/m$, we obtain $\Delta < 0$.

Proof of the sufficiency. If $\Delta < 0$, from (33) we have $P\{\lim_{n\to\infty}(1/n)\sum_{i=1}^{n}\xi_i < 0\} = 1$, which implies that $\exists\delta > 0, \exists K \in Z^+$, such that whenever $k \geq K$, $(1/k)\sum_{i=1}^{k}\ln|\lambda_i| < -\delta$, that is

$$\sum_{i=1}^{k}\ln|\lambda_i| < -k\delta. \tag{36}$$

Due to the arbitrariness of δ, we find that $\lim_{n\to\infty}\sum_{i=}^{k}\ln|\lambda_i| = -\infty$, which means that $P\{\lim_{n\to\infty}\rho_n = \lim_{n\to\infty}\prod_{i=1}^{n}\lambda_i = 0\} = 1$. This completes the proof of the sufficiency.

(2) From (33), we have the following equivalent propositions:

$$\begin{aligned}
\Delta = 0 &\Leftrightarrow P\left\{\lim_{n\to\infty}\frac{1}{n}\sum\nolimits_{i=1}^{n}\xi_i = 0\right\} = 1\\
&\Leftrightarrow \forall\varepsilon > 0, \exists K \in Z^+, \text{such that whenever } k \geq K,\ P\left\{\left|\frac{1}{k}\sum\nolimits_{i=1}^{k}\xi_i\right| < \varepsilon\right\} = 1\\
&\Leftrightarrow \forall\varepsilon > 0, \exists K \in Z^+, \text{such that whenever } k \geq K,\ P\{-k\varepsilon < \sum_{i=1}^{k}\xi_i < k\varepsilon\} = 1\\
&\Leftrightarrow P\{-\infty < \lim_{n\to\infty}\ln\rho_n < \infty\} \Leftrightarrow P\{0 < \lim_{n\to\infty}\rho_n < \infty\}.
\end{aligned} \tag{37}$$

Thus, considering the case for $\Delta < 0$ in (1) and the case for $\Delta = 0$, we find that the first proposition in (1) holds.

Similarly,

$$\begin{aligned}
\Delta > 0 &\Leftrightarrow P\left\{\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\xi_i > 0\right\} = 1\\
&\Leftrightarrow \exists\delta > 0, \quad \exists K \in Z^+, \text{such that whenever } k \geq K,\ P\left\{\frac{1}{k}\sum_{i=1}^{k}\zeta_i > \delta\right\}\\
&= 1, \text{i.e. } P\left\{\sum_{i=1}^{k}\zeta_i > k\delta\right\} = 1\\
&\Leftrightarrow P\left\{\lim_{n\to\infty}\sum_{i=1}^{k}\xi_i = \infty\right\} = 1 \Leftrightarrow P\left\{\lim_{n\to\infty}\ln\rho_n = +\infty\right\} = 1
\end{aligned} \tag{38}$$

Thus the second proposition in (2) holds.
This completes the proof of the second part of the theorem.

Theorem 4 *A sufficient condition for* $\rho_n = \prod_{i=1}^{n}\lambda_i$ *to be probabilistically bounded is that* $0 < \alpha < 1$ *and* $0 < \beta < 2$.

Proof Since $\{\lambda_n\}$ is a sequence of independent identically distributed (i.i.d.) random variables with each λ_n subject to the same normal distribution, i.e., $\lambda_n \sim N(1-\beta, \alpha^2)$, the expectation and the variance of $\rho_n = \prod_{i=1}^{n} \lambda_i$ can be given by

$$E[\rho_n] = E\left[\prod_{i=1}^{n} \lambda_i\right] = [E[\lambda_n]]^n = (1-\beta)^n, \tag{39}$$

and

$$Var[\rho_n] = E[(\rho_n)^2] - E[\rho_n]^2 = E\left[\prod_{i=1}^{n} \lambda_i^2\right]^n - E[\rho_n]^2 = [\alpha^2 + (1-\beta)^2]^n - (1-\beta)^{2n}. \tag{40}$$

A sufficient condition for ρ_n to converge is $E[\rho_n] \to 0$ and $Var[\rho_n] \to 0$ (i.e., mean square convergence of ρ_n), which implies that $0 < \beta < 2$ and $0 < \alpha < 1$. This completes the proof of the theorem. □

References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *12*, 281–305.

Bonyadi, M. R., Michalewicz, Z., & Li, X. (2014). An analysis of the velocity updating rule of the particle swarm optimization algorithm. *Journal of Heuristics*, *20*(4), 417–452.

Bratton, D., & Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Proceedings of IEEE swarm intelligence symposium*, (pp. 120–127). New York: IEEE Press.

Bull, A. (2011). Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, *12*, 2879–2904.

Cleghorn, W. C., & Engelbrecht, A. P. (2014). Particle swarm convergence: Standardized analysis and topological influence. Lecture Notes in Computer Science, *8867*, pp. 134–145.

Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of congress on evolutionary computation*. (pp. 1951–1957). New York: IEEE Press.

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(2), 58–73.

Courant, R. (1989). *Introduction to calculus and analysis*. Berlin: Springer.

Day, R. W., & Quinn, G. P. (1989). Comparisons of treatments after an analysis of variance in ecology. *Ecological Monographs*, *59*, 433–463.

Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, (pp. 39–43). New York: IEEE Press.

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 congress on evolutionary computation (CEC '00)*, *vol. 1*. (pp. 84-88). New York: IEEE Press.

Emara, H. M., & Fattah, H. A. A. (2004). Continuous swarm optimization technique with stability analysis. In *Proceedings of American control conference*, (pp. 2811–2817). New York: IEEE Press.

Engelbrecht, A. P. (2013). Particle swarm optimization: global best or local best? In *Proceedings of the 11th Brazilian congress on computational intelligence*, (pp. 124–135). New York: IEEE Press.

Escalante, H. J., Montes, M., & Sucar, L. E. (2009). Particle swarm model selection. *Journal of Machine Learning Research*, *10*, 405–440.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagne, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

Fournier, H., & Teytaud, O. (2011). Lower bounds for comparison based evolution strategies using VC-dimension and sign patterns. *Algorithmica*, *59*(3), 387–408.

Gavi, V., & Passino, K. M. (2003). Stability analysis of social foraging swarms. *IEEE Transactions on Systems, Man and Cybernetics*, *34*(1), 539–557.

Henning, P., & Keifel, M. (2013). Quasi-Newton methods: A new direction. *Journal of Machine Learning Research*, *14*, 843–865.

Jiang, M., Luo, Y. P., & Yang, S. Y. (2007). Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, *102*, 8–16.
Kadirkamanathan, V., Selvarajah, K., & Fleming, P. J. (2006). Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Transactions on Evolutionary Computation*, *10*(3), 245–255.
Kennedy, J. (1998). The behavior of particle. In *Proceedings of 7th annual conference on evolutionary programming*, (pp. 581–589). Berlin: Springer.
Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proceedings of congress on evolutionary computation*, (pp. 1931–1938). New York: IEEE Press.
Kennedy, J. (2002). Stereotyping: Improving particle swarm performance with cluster analysis. In *Proceedings of congress on computational intelligence*, (pp. 1671–1676). New York: IEEE Press.
Kennedy, J. (2003). Bare bones particle swarms. In *Proceedings of IEEE swarm intelligence symposium*, (pp. 80–87). New York: IEEE Press.
Kennedy, J. (2004). Probability and dynamics in the particle swarm. In *Proceedings of congress on evolutionary computation*, (pp. 340–347). New York: IEEE Press.
Kennedy, J. (2006). In search of the essential particle swarm. In *Proceedings of IEEE world congress on computational intelligence*, (pp. 1694–1701). New York: IEEE Press.
Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, (pp. 1942–1948). New York: IEEE Press.
Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of 2002 congress on evolutionary computation*, (pp. 1671–1676). New York: IEEE Press.
Kittel, C., & Kroemer, H. (1980). *Thermal physics* (2nd ed.). San Francisco: W.H. Freeman.
Krohling, R. A. (2004). Gaussian swarm: A novel particle swarm optimization algorithm. In *Proceedings of IEEE conference on cybernetics and intelligent systems*, (pp. 372–376). New York: IEEE Press.
Lane, J., Engelbrecht, A., & Gain, J. (2008). Particle swarm optimization with spatially meaningful neighbours. In *Proceedings of 2008 IEEE swarm intelligence symposium*, (pp. 1–8). New York: IEEE Press.
Li, X. (2004). Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In *Proceedings of 2004 genetic and evolutionary computation conference*, (pp. 105–116).
Li, X. (2010). Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation*, *14*(1), 150–169.
Liang, J. J., & Suganthan, P. N. (2005). Dynamic multiswarm particle swarm optimizer (DMS-PSO). In *Proceedings of IEEE swarm intelligence symposium*, (pp. 124–129). New York: IEEE Press.
Liang, J. J., Qin, A. K., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, *10*(3), 281–295.
Lu, Y., Wang, S., Li, S., & Zhou, C. (2011). Particle swarm optimizer for variable weighting in clustering high-dimensional data. *Machine Learning January*, *82*(1), 43–70.
Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, *8*(3), 204–210.
Omar, M. A. (1993). *Elementary solid state physics: Principles and applications*. Reading, MA: Addison-Wesley.
Ozcan, E., & Mohan, C. K. (1999). Particle swam optimization: Surfing the waves. In *Proceedings of 1999 IEEE congress on evolutionary computation*, (pp. 1939–1944). New York: IEEE Press.
Parrott, D., & Li, X. (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation. *IEEE Transactions on Evolutionary Computation*, *10*(4), 440–458.
Pelikan, M. (2012). Probabilistic model-building genetic algorithms. In *Proceedings of genetic and evolutionary computation conference*, (pp. 777–804).
Poli, R. (2007). An analysis of publications on particle swarm optimisation applications. *Technical report CSM-469*, Department of Computer Science, University of Essex, UK.
Poli, R., & Langdon, W. B. (2007). Markov chain models of bare-bones particle swarm optimizers. In *Proceedings of annual genetic and evolutionary computation conference*, (pp. 142–149).
Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, *2008*, 1–10.
Poli, R. (2009). Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. *IEEE Transactions on Evolutionary Computation*, *13*(4), 712–721.
Richer, T. J., & Blackwell, T. M. (2006). The Levy particle swarm. In *Proceedings of congress on evolutionary computation*, (pp. 808–815). New York: IEEE Press.
Secrest, B., & Lamont, G. (2003). Visualizing particle swarm optimization-gaussian particle swarm optimization. In *Proceedings of IEEE swarm intelligence symposium*, (pp. 198–204). New York: IEEE Press.

Shi, Y., & Eberhart, R. C. (1998a). A modified particle swarm optimizer. In *Proceedings of IEEE international conference on evolutionary computation*, (pp. 69–73). New York: IEEE Press.

Shi, Y., & Eberhart, R. C. (1998b). Parameter selection in particle swarm optimization. In *Proceedings of 7th conference on evolutionary programming VII (EP '98)*, (pp. 591–600). Berlin: Springer.

Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation (CEC '99), vol. 3*, (pp. 1945–1950). New York: IEEE Press.

Shiryayev, A. N. (1984). *Probability*. New York: Springer.

Solis, F. J., & Wets, R. J.-B. (1981). Minimization by random search techniques. *Mathematics of Operations Research*, *6*(1), 19–30.

Sra, S., Nowozin, S., & Wright, S. J. (2011). *Optimization for machine learning*. Cambridge: MIT Press.

Suganthan, P. N. (1999). Particle warm optimizer with neighborhood operator. In *Proceedings of congress on evolutionary computation*, (pp. 1958–1962). New York: IEEE Press.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 specialsession on real-parameter optimization. *Technical report*, Nanyang Technological University, Singapore, May 2005 and KanGAL Report #2005005, IIT Kanpur, India.

Sun, Q., Pfahringer, B., & Mayo, M. (2013). Towards a framework for designing full model selection and optimization systems. In *Proceedings of the 11th international workshop on multiple classifier systems (MCS'13)*, (pp. 259–270), Berlin: Springer.

Sun, J., Fang, W., Wu, X., Palade, V., & Xu, W. (2012b). Quantum-behaved particle swarm optimization: Analysis of the individual particle behavior and parameter selection. *Evolutionary Computation*, *20*(3), 349–393.

Sun, J., Palade, V., Cai, Y., Fang, W., & Wu, X. (2014a). Biochemical systems identification by a random drift particle swarm optimization approach. *BMC Bioinformatics*, *15*(Suppl 6), S1.

Sun, J., Palade, V., Wu, X., & Fang, W. (2014b). Multiple sequence alignment with hidden Markov models learned by random drift particle swarm optimization. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *11*(1), 243–257.

Sun, J., Zhao, J., Wu, X., Cai, Y., & Xu, W. (2010). Parameter estimation for chaotic systems with a drift particle swarm optimization method. *Physics Letters A*, *374*(28), 2816–2822.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, *85*(6), 317–325.

van den Bergh, F. (2002). An analysis of particle swarm optimizers. *Ph.D. dissertation*, University of Pretoria, Pretoria, South Africa.

van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 225–239.

Van den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences*, *176*(8), 937–971.

Van den Bergh, F., & Engelbrecht, A. P. (2010). A convergence proof for the particle swarm optimizer. *Fundamenta Informaticae*, *105*(4), 341–374.

Veeramachaneni, K., Wagner, M., & O'Reilly, U.-M., (2012). Frank Neumann: Optimizing energy output and layout costs for large wind farms using particle swarm optimization. In *Proceedings of IEEE congress on evolutionary computation*, (pp. 1–7). New York: IEEE Press.

Zhang, Y., Gong, D., Sun, X., & Geng, N. (2014). Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis. *Soft Computing*, *18*(7), 1337–1352.