Swarm Intell (2014) 8:159–198
DOI 10.1007/s11721-014-0095-1

A locally convergent rotationally invariant particle swarm optimization algorithm

Mohammad Reza Bonyadi · Zbigniew Michalewicz

Received: 13 August 2013 / Accepted: 2 June 2014 / Published online: 1 August 2014

Abstract Several well-studied issues in the particle swarm optimization algorithm are outlined and some earlier methods that address these issues are investigated from the theoretical and experimental points of view. These issues are the: stagnation of particles in some points in the search space, inability to change the value of one or more decision variables, poor performance when the swarm size is small, lack of guarantee to converge even to a local optimum (local optimizer), poor performance when the number of dimensions grows, and sensitivity of the algorithm to the rotation of the search space. The significance of each of these issues is discussed and it is argued that none of the particle swarm optimizers we are aware of can address all of these issues *at the same time*. To address all of these issues at the same time, a new general form of velocity update rule for the particle swarm optimization algorithm that contains a user-definable function f is proposed. It is proven that the proposed velocity update rule guarantees to address all of these issues if the function f satisfies the following two conditions: (i) the function f is designed in such a way that for any input vector $\vec{y}$ in the search space, there exists a region A which contains $\vec{y}$ and $f(\vec{y})$ can be located anywhere in A, and (ii) f is invariant under any affine transformation. An example of function f is provided that satisfies these conditions and its performance is examined through some experiments. The experiments confirm that the proposed algorithm (with an appropriate function f) can effectively address all of these issues at the same time. Also,

Electronic supplementary material The online version of this article (doi:10.1007/s11721-014-0095-1) contains supplementary material, which is available to authorized users.

M. R. Bonyadi (✉) · Z. Michalewicz
School of Computer Science, The University of Adelaide, Rm 4.52, Adelaide, SA 5005, Australia
e-mail: vardiar@gmail.com; mohammad.bonyadi@adelaide.edu.au; mrbonyadi@cs.adelaide.edu.au

Z. Michalewicz
Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland

Z. Michalewicz
Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland
e-mail: zbyszek@cs.adelaide.edu.au

comparisons with earlier methods show that the overall ability of the proposed method for solving benchmark functions is significantly better.

Keywords Particle swarm optimizer · Local convergence · Rotation invariance · Stagnation

1 Introduction

In this paper, we modify the particle swarm optimization (PSO) algorithm to enhance its performance for the following class of optimization problems:

$$\text{minimize } F(\vec{x}), \vec{x} = \langle x_1, \ldots, x_d \rangle \in S \subseteq R^d,$$

where d is the number of dimensions, and the search space S is defined as a hyper-cube in R^d: $l_i \leq x_i \leq u_i$ (l_i and u_i are the values of lower bound and upper bound of the ith dimension of the search space, respectively). In the rest of this paper, without losing generality, we consider minimization problems only.

PSO is a stochastic population-based optimization method that had its origin in the works by Kennedy and Eberhart (1995). It has been successfully applied to many problems such as artificial neural network training, function optimization, fuzzy system control, and pattern classification (Engelbrecht 2005; Poli 2008), to name but a few. Due to the ease of implementation and fast convergence to acceptable solutions, PSO has received more attention in recent years (Poli 2008). However, there are some issues in the original version of PSO introduced in Kennedy and Eberhart (1995) [as well as in its well-studied variant (Shi and Eberhart 1998a)] including: *stagnation* [getting stuck in some points in the search space without further movement (Van den Bergh and Engelbrecht 2010; Witt 2009; Van den Bergh and Engelbrecht 2002; Spears et al. 2010)], *dimensional stagnation* [particles oscillate along one of the dimensions (Van den Bergh and Engelbrecht 2010)], *swarm size* [poor performance when the swarm size is small (Van den Bergh and Engelbrecht 2010)][1], *local convergence* [no guarantee to converge to a local optimum (Van den Bergh and Engelbrecht 2002; Van den Bergh and Engelbrecht 2010; Schmitt and Wanka 2013)], *problem scale* [poor performance in improving the initial solution when the number of dimensions grows (Vesterstrom and Thomsen 2004; Zhao et al. 2008; Li and Yao 2011)][2], and *rotation variance* [sensitivity of the algorithm to the rotation of the search space (Wilke et al. 2007b; Spears et al. 2010)]. Many studies have been conducted so far to address one or two of these issues (Li and Yao 2011; Xinchao 2010; Wilke et al. 2007b; Van den Bergh and Engelbrecht 2002; Van den Bergh and Engelbrecht 2010; Schmitt and Wanka 2013; Bonyadi et al. 2013).

In this paper, these issues are divided into two groups: *primary* issues (stagnation, local convergence, and rotation variance) and *secondary* issues (dimensional stagnation, swarm

[1] Note that the swarm size issue should not be mixed with the concept of selecting the best population size for an algorithm. The swarm size issue means that the algorithm should be able to perform well even with a small number of particles. However, it does not mean that a larger population size does not affect the performance of the algorithm.

[2] Note that the definition of the "problem scale" issue should not be mixed with "large scale optimization." In large scale optimization the aim is to improve algorithms so that they are able to find quality solutions for problems with high number of dimensions. In the problem scale issue (discussed in this paper) we are after guaranteeing improvement of the initial solutions by the algorithm when the number of dimensions grows. The presence of the problem scale issue implies that the algorithm is not a good method for large scale optimization. However, a good performance in large scale optimization by an algorithm implies that there is no problem scale issue in that algorithm.

size, and problem scale), based on dependencies among them. We show that if the primary issues are addressed then the secondary issues are addressed as well (see Sect. 3.7). Thus, the main focus of this paper is to address the primary issues. Also, a new general form of velocity update rule for PSO is proposed. This new general form of velocity update rule contains a user-definable function f. It is proven that if the function f satisfies two specific conditions (see Sect. 4.1) then the proposed method guarantees to address all primary issues *at the same time*. A particular model is derived from the proposed general model in which a specific function f is designed that satisfies these two conditions. This particular model is applied to some benchmark optimization functions, and its results are discussed.

The rest of the paper is organized as follows. After a brief review of the original formulation of PSO in Sect. 2.1, the role of random matrices in the trajectory of particles is explained in Sect. 2.2. Some variants of PSO are outlined in Sect. 2.3. In Sect. 3, some issues in PSO are inspected in detail and the abilities of the introduced variants in Sect. 2.3 to address these issues are analyzed. In Sect. 4, a new general form of the velocity update rule for PSO is proposed. Also, in that section the ability of the proposed method to address the introduced issues in Sect. 3 are investigated in detail. In Sect. 5, the proposed method is compared to some other PSO variants and its performance is analyzed. The last section concludes by summing up the advantages and disadvantages of the proposed method and discusses some potential future work.

2 Background

Some basic information about PSO including its original formulation, role of random matrices in that formulation, and some of the variants of the algorithm are discussed in this section. The PSO variants for the overview have been selected based on their contributions in addressing the issues that this paper investigates. These methods are investigated further in Sect. 3 to evaluate their abilities to address primary and secondary issues.

2.1 Basics of PSO

The PSO (Kennedy and Eberhart 1995) algorithm is based on a population (referred to as *swarm*) of $n > 1$ particles (swarm size n); each particle is defined by three d-dimensional vectors:

- *Position* ($\vec{x}_t^i$)—is the position of the ith particle in the tth iteration. This is used to evaluate the particle's quality.
- *Velocity* ($\vec{V}_t^i$)—is the direction and length of movement of the ith particle in the tth iteration.
- *Personal best* ($\vec{p}_t^i$)—is the best position that the ith particle has visited in its lifetime (up to the tth iteration).

All of these vectors are updated at every iteration t for each particle i:

$$\vec{V}_{t+1}^i = \mu\left(\vec{x}_t^i,\ \vec{V}_t^i,\ N_t^i\right),\ \text{for } i = 1, \ldots,\ n, \tag{1a}$$

$$\vec{x}_{t+1}^i = \xi\left(\vec{x}_t^i,\ \vec{V}_{t+1}^i\right),\ \text{for } i = 1, \ldots,\ n, \tag{1b}$$

$$\vec{p}_{t+1}^i = \begin{cases} \vec{p}_t^i & \text{if } F\left(\vec{p}_t^i\right) < F(\vec{x}_{t+1}^i) \\ \vec{x}_{t+1}^i & \text{otherwise} \end{cases},\quad \text{for } i = 1, \ldots,\ n. \tag{1c}$$

In Eq. (1a), N_t^i, known as the neighbor set of particle i, is a subset of personal best positions of the particles which contribute to the velocity update rule of particle i at iteration t, i.e., $N_t^i = \left\{\vec{p}_t^k | k \in \left\{T_t^i \subseteq \{1, 2, \ldots, n\}\right\}\right\}$ where T_t^i is a set of indices of particles which contribute to the velocity update rule of particle i at iteration t. Clearly, the strategy of determining T_t^i might be different for various types of PSO algorithms and it is usually referred to as the *topology* of the swarm. Many different topologies have been proposed so far (Mendes et al. 2004), e.g., global best topology, ring topology, cluster topology, pyramid typology, each of them presenting advantages and disadvantages (Mendes et al. 2004; Clerc 2006). The function $\mu(.)$ calculates the new velocity vector for particle i according to its current position, current velocity $\vec{V}_t^i$, and neighbor set N_t^i. In Eq. (1b), $\xi(.)$ is a function which calculates the new position of particle i according to its previous position and its new velocity. Usually $\xi\left(\vec{x}_t^i,\ \vec{V}_{t+1}^i\right) = \vec{x}_t^i + \vec{V}_{t+1}^i$ is accepted for updating the position of particle i. In Eq. (1c), the new personal best position for particle i is updated according to the objective value (the value of the function F) of its previous personal best position and the current position. In the rest of this paper, these usual forms for the position update rule (Eq. 1b) and personal best updating rule (Eq. 1c) are assumed. In PSO, three updating rules (Eqs. 1a–1c) are applied to all particles iteratively until a predefined termination criterion, e.g., the maximum number of iterations or deviation from the global optimum (if known), is met. Also, $\vec{x}_0^i$ and $\vec{V}_0^i$ are generated either randomly or by using a heuristic method. Moreover, $\vec{p}_0^i$ is initialized to $\vec{x}_0^i$ for all particles.

In the original form of PSO (Kennedy and Eberhart 1995), the set T_t^i contained only two indices that were $\{i, \tau_t\}$ where $\tau_t = \underset{l \in \{1,\ldots,n\}}{\operatorname{argmin}} \{F\left(\vec{p}_t^l\right)\}$. This topology is called the *global best topology* for PSO. The particle τ_t is referred to as the *global best particle* and the personal best of the particle τ_t is called the *global best vector* ($\vec{g}_t$). For the original PSO, the function $\mu(.)$ in Eq. 1a was defined (Kennedy and Eberhart 1995) as:

$$\vec{V}_{t+1}^i = \vec{V}_t^i + \varphi_1 R_{1t} \underbrace{\left(\vec{p}_t^i - \vec{x}_t^i\right)}_{\substack{\text{Personal} \\ \text{Influence}(\overrightarrow{PI})}} + \varphi_2 R_{2t} \underbrace{\left(\vec{g}_t - \vec{x}_t^i\right)}_{\substack{\text{Social} \\ \text{Influence}(\overrightarrow{SI})}}, \tag{2}$$

where φ_1 and φ_2 are two real numbers called acceleration coefficients and $\vec{p}_t^i$ and $\vec{g}_t$ are the personal best (of the particle i) and the global best vectors, respectively, at iteration t. The acceleration coefficients control the effect of personal and global best vectors on the movement of particles and play an important role in the convergence of the algorithm (Van den Bergh and Engelbrecht 2006; Clerc and Kennedy 2002). The role of vectors $\overrightarrow{PI} = \vec{p}_t^i - \vec{x}_t^i$ (Personal Influence) and $\overrightarrow{SI} = \vec{g}_t - \vec{x}_t^i$ (Social Influence) is to *attract* the particles to move toward known quality solutions, i.e., personal and global best vectors. R_{1t} and R_{2t} are two $d \times d$ diagonal matrices[3] (Montes de Oca et al. 2009; Clerc 2006), whose elements are random numbers distributed uniformly ($\sim U(0, 1)$) in the interval [0, 1]. Note that matrices R_{1t} and R_{2t} are generated at each iteration for each particle separately.

In some PSO variants (e.g., Shi and Eberhart 1998b), the value of the velocity is restricted to the range $[-V_{\max}, V_{\max}]$ (if $\vec{V}_{t+1}^i > V_{\max}$ then $\vec{V}_{t+1}^i = V_{\max}$ and if $\vec{V}_{t+1}^i < -V_{\max}$ then $\vec{V}_{t+1}^i = -V_{\max}$), where $V_{\max}$ is the maximum allowed velocity. This restriction is applied to the velocity of the particles to limit the step size of their movements. Also, if a particle leaves the search space then its objective value is not considered for updating the personal

[3] Alternatively, these two random matrices are often considered as two random vectors. In this case, the multiplication of these random vectors by $\overrightarrow{PI}$ and $\overrightarrow{SI}$ is element-wise.

and global best vectors. See Helwig and Wanka (2007, 2008) and Engelbrecht (2012) for other studies on movement of particles out of the search space boundaries.

In 1998, (Shi and Eberhart 1998a) introduced a new coefficient ω (known as the *inertia weight*) to control the influence of the last velocity value on the updated velocity. Indeed, Eq. 2 was rewritten as:

$$\vec{V}_{t+1}^{i} = \omega \vec{V}_{t}^{i} + \varphi_1 R_{1t}(\vec{p}_{t}^{i} - \vec{x}_{t}^{i}) + \varphi_2 R_{2t}(\vec{g}_{t} - \vec{x}_{t}^{i}). \tag{3}$$

The coefficient ω controls the influence of the previous velocity on movement. Throughout the paper, this variant is referred to as the standard PSO, SPSO. The iterative application of Eq. 3 (plus the position update rule) causes the particles to oscillate around personal and global best vectors (Poli et al. 2007; Clerc and Kennedy 2002). This oscillation is controlled by three parameters: ω, φ_1, and φ_2, so that the larger ω is with respect to φ_1 and φ_2, the more explorative the particles are, and vice versa; for the smaller ω there is a larger tendency for particles to exploit solutions around global and personal best vectors. If these coefficients are not set accurately, the velocity vector might increase unboundedly (this is called "swarm explosion") and particles move to infinity (Clerc and Kennedy 2002). There have been some studies on the stability of the particles, i.e., convergence of the particles to a point (called equilibrium point) (Clerc and Kennedy 2002; Trelea 2003; Jiang et al. 2007a; Jiang et al. 2007b; Poli 2009; Van den Bergh and Engelbrecht 2006; Bonyadi and Michalewicz 2014) to prevent the swarm explosion. It was found (Trelea 2003; Jiang et al. 2007a; Van den Bergh and Engelbrecht 2006) that, if the value of coefficients are selected within a specific boundary then particles converge to their equilibrium point (this boundary for coefficients is called *convergence boundary*). Also, it was proven that at the equilibrium point, $\vec{V}_{t}^{i} = 0$ and $\vec{g}_{t} = \vec{x}_{t}^{i} = \vec{p}_{t}^{i}$ (Poli 2009). In fact, particles do not stop searching until their personal best becomes the global best of the swarm and their velocity becomes zero. Note that this condition ($\vec{V}_{t}^{i} = 0$ and $\vec{g}_{t} = \vec{x}_{t}^{i} = \vec{p}_{t}^{i}$) is necessary for the convergence to an equilibrium point.

In the remaining of this section, some background information on the role of the random matrices (R_{1t} and R_{2t}) in SPSO is provided Sect. 2.2 and some variants of PSO Sect. 2.3 are reviewed.

2.2 Role of the random matrices in SPSO

The main task of the random diagonal matrices (R_{1t} and R_{2t} in Eqs. 2 and 3) is to apply perturbation to the movement vector, i.e., the velocity vector, of particles in SPSO. Any such perturbation of the movement vector contains two parts: *magnitude perturbation* and *direction perturbation*. The magnitude perturbation diversifies the step size of particles movement (Wilke et al. 2007a) (particles do not move with fixed step sizes), whereas the direction perturbation guarantees the direction diversity in the swarm (particles do not move following a fixed direction); these perturbations are important for the exploitation and exploration ability of the algorithm as was studied by Wilke et al. (2007a). Note that if a small magnitude perturbation with large direction perturbation is applied to a particle then the particle behaves more exploitatively. However, if the magnitude perturbation is large then the particle behaves more exploratively. Figure 1 shows the perturbation that random matrices apply to $\overrightarrow{SI}$ and $\overrightarrow{PI}$ in SPSO.

The gray rectangles in Fig. 1 indicate the area that the vectors $R_{2t}\overrightarrow{SI}$ and $R_{1t}\overrightarrow{PI}$ might be in. It is obvious that after multiplying $\overrightarrow{SI}$ and $\overrightarrow{PI}$ by the random matrices the direction and magnitude of the new vectors are different from the original vectors (Bonyadi et al.

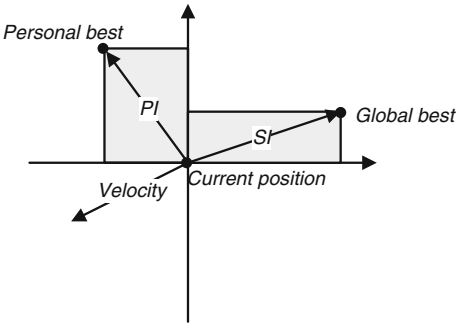

Fig. 1 R_{1t} and R_{2t} apply perturbation on $\overrightarrow{PI}$ and $\overrightarrow{SI}$, respectively. The area that the $R_{2t}\overrightarrow{SI}$ and $R_{1t}\overrightarrow{PI}$ might be in is shown in *gray*

2014). These perturbations prevent particles from moving exactly towards the known quality solutions ($\vec{p}_t^{\,i}$ and $\vec{g}_t$); rather, they enable particles to conduct exploitation around these quality solutions. If particles constantly move towards quality known solutions ($\vec{p}_t^{\,i}$ and $\vec{g}_t$) without any perturbation, all particles would collapse on these solutions at the early stage of the optimization process, that is not desirable (Wilke et al. 2007a). Thus, the role of random matrices is important for the exploration and exploitation abilities of the algorithm (Wilke et al. 2007b, Wilke et al. 2007a).

2.3 Some PSO variants

There are numerous PSO variants that have been proposed over the last 15 years; they are based on different equations for updating velocities/positions (Ghosh et al. 2010; Liang et al. 2006), different topologies (Mendes et al. 2004; Xinchao 2010), different values for various parameters (Clerc and Kennedy 2002; Ratnaweera et al. 2004), different hybridizations (Huang et al. 2010; Wang et al. 2011), and population sizing (Hsieh et al. 2009; Chen and Zhao 2009). All these variants aimed at improving the searching ability of SPSO or other PSO variants (see Tu and Lu 2004; Poli et al. 2007). In this subsection some of these PSO variants are presented. These methods have been selected because they have tried to address the issues that this paper investigates. The variants are presented in chronological order. These variants are revisited in Sect. 3 and their performance in addressing related issues in PSO is investigated in detail.

In 2002, a PSO variant, called COnstriction coefficient PSO (COPSO), was proposed (Clerc and Kennedy 2002). The velocity update rule in COPSO (given below) is different from the one in SPSO:

$$\vec{V}_{t+1}^{\,i} = \chi\left(\vec{V}_t^{\,i} + c_1 R_{1t}\left(\vec{p}_t^{\,i} - \vec{x}_t^{\,i}\right) + c_2 R_{2t}\left(\vec{g}_t - \vec{x}_t^{\,i}\right)\right). \tag{4}$$

The parameter χ is called *constriction factor*. The authors demonstrated that tuning the values of χ, c_1 and c_2 can prevent the swarm explosion and can lead to better exploration within the search space. It was stated (Clerc and Kennedy 2002) that the parameters (χ, c_1 and c_2) should be set so that the following equation is satisfied:

$$\chi = \frac{2z}{\left|2 - c - \sqrt{c^2 - 4c}\right|}, \tag{5}$$

where $c = c_1 + c_2 > 4$ and $z \in [0, 1]$.

Another PSO variant called Guaranteed Convergence PSO (GCPSO) was proposed (Van den Bergh and Engelbrecht 2002). In that variant, the velocity update rule for all particles

was the same as the one in COPSO except for the particle τ_t (recall that the particle τ_t is the particle in which $\vec{p}_t^{\,i} = \vec{g}_t$). The velocity update rule for the GCPSO was as follows:

$$\vec{V}_{t+1}^{\,i} = \begin{cases} -\vec{x}_t^{\,i} + \vec{g}_t + \omega \vec{V}_t^{\,i} + \vec{\rho} & \text{if } i = \tau_t \\ \omega \vec{V}_t^{\,i} + \varphi_1 R_t^i \left(\vec{p}_t^{\,i} - \vec{x}_t^{\,i}\right) + \varphi_2 R_t^i \left(\vec{g}_t - \vec{x}_t^{\,i}\right) & \text{otherwise,} \end{cases} \tag{6}$$

where the values of all elements (dimensions) in $\vec{\rho}$ are equal to ρ, ρ is a non-zero random number. The vector $\vec{\rho}$ applies perturbation to prevent stagnation. In that paper, the value of ρ was determined through an adaptive approach at each iteration. Using Eq. (6), the movement of particle τ_t is guaranteed because ρ is determined randomly at each iteration, which guarantees non-zero value for $\vec{V}_{t+1}^{\,\tau_t}$ for any t.

Two years later, a general form of COPSO called the fully informed particle swarm (FIPS) was proposed (Mendes et al. 2004). In that method, particles were able to use information from all other particles in the swarm rather than only the global best particle. The velocity update rule for FIPS was as follows:

$$\vec{V}_{t+1}^{\,i} = \omega \vec{V}_t^{\,i} + \sum_{k \in T_t^i} \varphi_k R_{kt}^i \left(\vec{p}_t^{\,k} - \vec{x}_t^{\,i}\right). \tag{7}$$

In FIPS, several different topologies (sets T_t^i) were tested. Experiments showed that one of the most successful topologies in solving the tested benchmark functions was the *ring* topology. In the ring topology, each particle uses the best experience of two other particles (known as local best particle (*lb*)) in its neighbor together with its own experience (Mendes et al. 2004). Also, in all tested topologies the personal best vector of each particle contributed in its own velocity vector, i.e., for each particle i, $i \in T_t^i$. The ring topology is used for FIPS in this paper because this topology has offered best results reported in Mendes et al. (2004) for the algorithm.

In 2007, the usage of the random rotation matrices rather than the random diagonal matrices was proposed (Wilke et al. 2007b) and the resulting algorithm was called Rotation PSO (RPSO). In that method the velocity vector of each particle was multiplied by a random rotation matrix at each iteration. This rotation could guarantee direction diversity throughout the swarm. Also, the magnitude diversity was generated by multiplying a random scalar with the velocity vector. Experiments showed that the new method performs better than SPSO in solving some standard benchmark optimization functions. However, the time complexity of generating random rotation matrices had higher order in comparison to generating random diagonal matrices ($O(d^2)$ to generate random rotation matrices versus $O(d)$ to generate random diagonal matrices). This causes RPSO to become slower in a quadratic order when the number of dimensions grows.

The velocity update rule of COPSO was further revised in Xinchao (2010), where the authors proposed a method called perturbed particle swarm algorithm (pPSA). In pPSA, the vector $\vec{g}_t$ in the velocity update rule of SPSO was substituted by $N(\vec{g}_t,\ \sigma^2 I)$, where N is the normal distribution and σ is the standard deviation. The velocity update rule was revised as follows:

$$\vec{V}_{t+1}^{\,i} = \omega \vec{V}_t^{\,i} + \varphi_1 R_{1t}^i \left(\vec{p}_t^{\,i} - \vec{x}_t^{\,i}\right) + \varphi_2 R_{2t}^i \left(\vec{g}'_t - \vec{x}_t^{\,i}\right), \tag{8}$$

where $\vec{g}'_t = N\left(\vec{g}_t, \sigma^2 I\right)$ and σ was a real number indicating the standard deviation of the distribution. Definitions for R_{1t}^i and R_{2t}^i were the same as that of SPSO (randomly generated diagonal matrices). The value of σ is either $\sigma_{\min}$ or $\sigma_{\max}$ and is switched after a certain number of iterations; for the first 50 percent of iterations $\sigma = \sigma_{\max} = 0.15$ and for the last 50 percent, $\sigma = \sigma_{\min} = 0.001$. The algorithm was applied to some standard unimodal/multimodal

optimization benchmarks. Results showed that the algorithm performs better than SPSO in terms of the quality of solutions and of robustness.

Another variant of PSO was introduced in Li and Yao (2011). In this method, the position of particle i in the jth dimension is determined by applying the following formulae:

$$x_{t+1}^{i,j} = \begin{cases} p_t^{i,j} + C(1)|p_t^{i,j} - lb_t^{i,j}| & \text{if rand} < q \\ lb_t^{i,j} + N(0,1)|p_t^{i,j} - lb_t^{i,j}| & \text{otherwise} \end{cases} \quad (9)$$

where $C(1)$ is a random number generated following the Cauchy distribution, $N(0, 1)$ is a random number generated following the normal distribution with mean 0 and variance 1, q is a user specified value and rand generates a uniform random value in the interval [0, 1]. Also, lb^i is the best particle in the neighbor of particle i. This method was called Cauchy-Gaussian PSO (UGPSO) which belongs to a class of PSO variants known as Bare-Bones PSO (Kennedy 2003). UGPSO was combined with a cooperation coevolution (Potter and De Jong 1994) approach to enable the algorithm to perform better on large scale problems.

3 Some issues in PSO and their significance

In this section, some issues in SPSO are investigated and existing methods for addressing these issues are discussed. These methods are the ones that were overviewed in Sect. 2.

The main reason behind the emergence of several variants of PSO was the need to address some issues identified in the formulation of SPSO, e.g., stagnation (Van den Bergh and Engelbrecht 2002, 2010; Witt 2009), rotation variance (Wilke et al. 2007a, b; Wilke 2005), swarm size issue (Van den Bergh and Engelbrecht 2002, 2010), biases (Spears et al. 2010), and large scale optimization (Li and Yao 2011; Cheng et al. 2011). The issues in SPSO (six issues) that are investigated in this paper are presented in detail in the following Sect. 3.1–3.6. Each of these subsections contains: an introduction to the issue, a discussion on the significance of the issue, and an investigation of the ability of the methods outlined in Sect. 2 in addressing that issue. At the end of this Sect. 3.7 the dependencies among these issues are discussed.

Note that there are some methods which address these issues in SPSO at the meta-algorithmic level, e.g., hybridization of the algorithm to address problem scale (Van den Bergh and Engelbrecht 2004); restarting particles to address stagnation (Garcia-Nieto and Alba 2011); and changing the population size if the algorithm does not work with the current number of individuals (Hsieh et al. 2009). However, the main focus of this paper is to understand the reasons behind these issues in SPSO and how we can revise the algorithm to address them. This in fact results in the designation of a comprehensive method which is able to work in different environments with a variety of parameters. Thus, we only concentrate on the methods which have analyzed these issues at the algorithmic level.

3.1 Stagnation

In SPSO, if the current position of each particle is equal to its personal best and global best vectors and the velocity of all particles is zero, i.e., $\vec{x}_t^i = \vec{p}_t^i = \vec{g}_t$[4] and $\vec{V}_t^i = 0$ for all

[4] Note that in continuous space, the probability of hitting a particular point is zero. Thus, the probability of $\vec{p}_t^i = \vec{g}_t$ or $\vec{x}_t^i = \vec{g}_t$ or any other equality like $\vec{x}_t^i = \vec{p}_t^k$ is zero. However, because of the representation of the floating points on computers, the probability of this situation is non-zero. In addition, the particles converge to their equilibrium point if their coefficients are set within the convergence boundary. As in the equilibrium point

particles i, then all particles stop moving and no further improvement can take place (Van den Bergh and Engelbrecht 2002); this is called *stagnation* issue throughout this paper. Note that if the coefficients of the velocity vector are selected from the convergence boundary then particles converge to their equilibrium (see Sect. 2.1). However, there is no guarantee that this equilibrium point is a local optimizer of the objective function. Thus, it would be better to search the points around this equilibrium point for better solutions rather than stop moving. A general form of stagnation is defined in this paper as

$$\exists t \forall i \in \{1, 2, \ldots, n\} \forall k \in T_t^i \vec{x}_t^i = \vec{p}_t^k \quad \text{and} \quad \vec{V}_t^i = 0, \tag{10}$$

where T_t^i is the set of indices of all particles which contribute to the velocity update rule of particle i. Note that this general form is equivalent to stagnation as introduced in Van den Bergh and Engelbrecht (2002) if $T_t^i = \{i, \tau_t\}$. The significance of addressing stagnation is that as long as this issue exists, there is no guarantee that the algorithm can even improve the initial solution.

The stagnation issue was addressed in Van den Bergh and Engelbrecht (2002) [further investigated in Van den Bergh and Engelbrecht (2010)] and Xinchao (2010), where the authors proposed GCPSO and pPSA, respectively. In GCPSO, a new velocity update rule was introduced for the particle τ_t, in which a random location around $\vec{g}_t$ was generated and the particle τ_t (global best particle) was moved to this location in each iteration (see Eq. 6). The velocity rule for particle τ_t in GCPSO forced this particle to move in any situation with the hope of finding better solutions and updating $\vec{g}_t$. If movement of the particle τ_t leads to updating $\vec{g}_t$ then other particles jump out of the stagnation situation as well. In pPSA, in order to address the stagnation issue, a perturbed global best position (shown by $\vec{g}'_t$), rather than the original global best position ($\vec{g}_t$), was used in the velocity update rule for all particles. In fact, in each iteration t and for each particle i, a new position $\vec{g}'_t$ was randomly generated around $\vec{g}_t$ and used for calculating $\vec{V}_{t+1}^i$. Thus, even if $\vec{x}_t^i = \vec{p}_t^i = \vec{g}_t$ and $\vec{V}_t^i = 0$ for all particles in a particular iteration t, $\vec{V}_{t+1}^i$ is nonzero with nonzero probability due to the usage of $\vec{g}'_t$ rather than $\vec{g}_t$.

Clearly, the stagnation can happen in COPSO and FIPS. In FIPS, there is no strategy in the algorithm to move the particles after stagnation. Stagnation can happen also in UGPSO because, if all particles collapse to a single point then $p^{ij} = lb^{ij}$ for all particles i and dimensions j (see Eq. 10). Thus, the value of $|p_t^{i,j} - lb_t^{i,j}|$ becomes zero for all particles and dimensions. In this situation, there is no chance for the particles to move and they cannot be further improved.

3.2 Dimensional stagnation

A special case of stagnation happens when stagnation takes place only in one dimension. This special case is called *dimensional stagnation* (Clerc 2006; Van den Bergh and Engelbrecht 2010; Bonyadi et al. 2014).

$$\exists t \exists j \in \{1, 2, \ldots, d\} \forall i \in \{1, 2, \ldots, n\} \forall k \in T_t^i x_t^{ij} = p_t^{kj} \quad \text{and} \quad V_t^{ij} = 0, \tag{11}$$

where T_t^i is the set of indices of all particles which contribute to the velocity update rule of particle i. Clearly, if dimensional stagnation happens for all dimensions then Eq. 11 becomes

$\vec{V}_t^i = 0$ and $\vec{x}_t^i = \vec{p}_t^i = \vec{g}_t$, the stagnation happens for the particles at the equilibrium point. However, there is no guarantee that the equilibrium point is a local minimizer of the objective function. Also, the convergence to an equilibrium point is independent from the starting position of the particles, which means that restarting particles cannot guarantee avoidance of stagnation.

the same as Eq. 10. Addressing dimensional stagnation is important because as long as this issue exists, the particles might oscillate along one dimension and miss the global/local optimum. This has also been discussed in Schmitt and Wanka (2013) and Bonyadi et al. (2013) for SPSO.

In RPSO (Wilke et al. 2007b), the dimensional stagnation issue has been addressed by applying direction perturbation (using rotation matrices) to the velocity vector of the particles in each iteration. This guarantees that, even if $x_t^{ij} = p_t^{ij} = g_t^j$ and V_t^{ij} is 0 for a particle i and a dimension j, V_{t+1}^{ij} may have a nonzero value due to the rotation that is applied to the velocity vector. Nevertheless, RPSO still does not guarantee to address stagnation. In fact, when $\vec{x}_t^i = \vec{p}_t^i = \vec{g}_t$ and $\vec{V}_t^i = 0$, the length of $\vec{V}_{t+1}^i$ becomes zero and no rotation can take place on the velocity.

The issue of dimensional stagnation has not been addressed in FIPS, COPSO, and UGPSO. For example, in UGPSO, if $p^{ij} = lb^{ij}$ for a specific dimension j, the algorithm is in the dimensional stagnation situation.

3.3 Swarm size

In SPSO, particles may stop moving in the early stages of the optimization process if the swarm size is small (Van den Bergh and Engelbrecht 2010) (this is called *swarm size* issue in this paper). Although a particle swarm benefits from a group of particles cooperating with each other to solve a problem, it is sometimes beneficial to use a small swarm size. Examples of situations where a small swarm size is preferable include: cooperative coevolution with PSO (Van den Bergh and Engelbrecht 2004, 2001) and large scale optimization (Li and Yao 2011). Also, a small swarm size is more desirable in multi-agent search strategies or for problems with an expensive evaluation function (see Malan and Engelbrecht 2008; Engelbrecht 2011) for a discussion on the importance of swarm size).

GCPSO and COPSO were applied to some standard benchmark test functions with a small swarm size (Van den Bergh and Engelbrecht 2010). Experiments showed that GCPSO was able to find acceptable solutions even with a small swarm size. However, COPSO performed poorly when the number of particles was small ($n = 2$). In this paper, the ability of PSO variants, outlined in Sect. 2.3, in dealing with the swam size issue is examined through some experiments (see Sect. 5.3).

3.4 Local convergence

We define a local minimizer of objective function F over the search space S as follows: c_i is a local minimizer if there exists an open set $I_i \subseteq S$ (c_i cannot be on the borders of I_i) such that $\forall x \in I_i$, $F(c_i) \leq F(x)$. We define $R_{\varepsilon i}$ for each c_i as

$$R_{\varepsilon i} = \{x \epsilon I_i : F(x) < F(c_i) + \varepsilon\},$$

where ε is an arbitrarily small positive value. The optimality region of the objective function F is defined as $R_\varepsilon = \bigcup_i R_{\varepsilon i}$. The aim of a local search algorithm is to find a point in the search space that is within the optimality region (Van den Bergh and Engelbrecht 2010; Solis and Wets 1981).

A PSO method is locally convergent if

$$\forall i \lim_{t \to \infty} P\left(\vec{p}_t^i \in R_\varepsilon\right) = 1,$$

that is, for each particle i, the probability that the personal best of that particle, $\vec{p}_t^{\,i}$, is in the optimality region R_ε approaches 1 when the iteration number t approaches infinity[5].

Some optimization algorithms are not locally convergent, i.e., they cannot guarantee to find a local optimum in the search space. This is called *local convergence* issue. It is important to address the local convergence issue in PSO because as long as this issue exists, the algorithm might be unable to find better solutions close to the global best vector. In other words, the gradient of the objective function at $\vec{g}_t$ is non-zero while the algorithm has been stagnating. However, when the gradient at the point $\vec{g}_t$ is non-zero, it is expected that the optimization algorithm finds better solutions than the current $\vec{g}_t$. The local convergence property is also desirable in niching where it is required to find multiple local optima in the search space.

The stagnation situation can serve as an example to show that COPSO, RPSO, UGPSO, and FIPS did not address the local convergence issue. In fact, if these methods stick in the stagnation situation, the probability of movement of particles is zero, no matter whether any of the particles is in the optimality region. However, this issue does not exist in GCPSO because of the perturbation component ρ. Also, pPSA is able to locate a local optimum because the normal distribution around the global best vector guarantees movement of particles at every iteration.

Note also that global convergence has been studied for PSO variants (Hao and Wenbo 2011; Van den Bergh and Engelbrecht 2010). However, this topic is not in the scope of this study and it is left for future research.

3.5 Problem scale

Another well-studied issue of SPSO is that the performance of the algorithm is radically impaired when the problem scale grows (Vesterstrom and Thomsen 2004; Van den Bergh and Engelbrecht 2004). In fact, when the algorithm is applied to a large scale problem (we consider a problem to be *large scale* in our experiments if the number of dimensions is larger than 100) and it is not initialized to local minima of the objective function, particles are barely improved and sometimes are not improved at all (see Sect. 4.3 for detailed experiments on this issue). This issue is called *problem scale* issue throughout the paper.

The experimental results in Vesterstrom and Thomsen (2004) showed that SPSO is defeated by differential evolution and evolutionary strategies in large scale optimization (problems larger than 100 dimensions in those experiments). Addressing the problem scale issue is important as there are many optimization problems that involve many dimensions, e.g., in oil industry, chemistry, and physics. Note that the performance of SPSO can be improved at the meta-algorithmic level, e.g., by adding a local search method to the algorithm (Montes de Oca et al. 2011) or by restarting the particles (Garcia-Nieto and Alba 2011). However, the aim of this study is to investigate the reasons behind this issue and propose solutions at the algorithmic level.

To the best of our knowledge, the problem scale issue is not addressed in RPSO, COPSO, FIPS, pPSA, UGPSO, and GCPSO. In fact, there are no available results of applying these methods in their original form (without any hybridization or external processes, like the ones in Li and Yao (2011) and Van den Bergh and Engelbrecht (2004)) to large scale problems.

[5] Note that, as local convergence is a property of an algorithm, it is sufficient if the final solution presented by the algorithm is in the optimality region. As the final solution generated by a PSO algorithm is (usually) the global best vector, a PSO algorithm is locally convergent if the condition $\lim_{t\to\infty} P\,(\vec{g}_t \in R_\varepsilon) = 1$ is satisfied. However, if a PSO method satisfies this condition, there is only a guarantee that one particle (the global best particle) will converge to the optimality region. Thus, the other particles might converge to any other points in the search space that are not in the optimality region.

In order to investigate the ability of these methods in dealing with the problem scale issue, all of them are examined through experiments in this paper (see Sect. 5.3).

3.6 Rotation variance

The performance of SPSO is changed if the search space is rotated (Wilke et al. 2007b; Spears et al. 2010; Wilke 2005); this is called the *rotation variance* issue throughout the paper. Rotation of the search space of a problem usually causes the dimensions of the problem to become non-separable (the optimum solution cannot be located by optimizing the objective function along each dimension separately). As many optimization functions in real-world are non-separable, it is important that the optimization algorithm performs well when optimizing non-separable functions (see also Hansen et al. 2011). Addressing the rotation variance issue is desirable. Indeed, if an algorithm is rotation variant, it should be applied to a problem with a rotated search space to find out if any better solution under different rotations is achieved. However, this process is time consuming, especially for large scale optimization problems.

It was proven (Wilke et al. 2007b) that SPSO is rotationally variant and that RPSO is rotationally invariant. In order to confirm the theoretical investigations, the authors applied RPSO and SPSO to some benchmark test functions. These experiments confirmed that RPSO is rotationally invariant while SPSO is rotationally variant. RPSO was further investigated in Bonyadi et al. (2014). The rotation variance issue was also investigated in Spears et al. (2010). There are no results available to indicate whether other PSO variants, such as GCPSO, FIPS, pPSA, are rotation variant or not. In this paper, all of these methods are tested using the Ellipse function, proposed in Spears et al. (2010). This function has been designed to test the ability of optimization methods in addressing the rotation variance issue (see Sect. 4.3). Note that finding one example is sufficient to show that a PSO variant is not rotation invariant. However, a mathematical proof is needed to guarantee that a PSO variant is rotation invariant. Also, we have some experiments in Sect. 5.7 to evaluate the ability of some of these variants in addressing the rotation variance issue.

3.7 Relations between the issues

The discussed issues (stagnation, dimensional stagnation, swarm size, local convergence, problem scale, and rotation invariance) are not independent of each other, i.e., addressing one issue may offer a solution for another. Hence, in this paper these issues are categorized into two groups: *primary* and *secondary*. Primary issues are the issues that are not addressed even if any other issue is resolved. The remaining issues, the issues that are addressed if primary issues are resolved, are called secondary issues.

For example, it is obvious that addressing stagnation resolves dimensional stagnation. However, addressing dimensional stagnation does not resolve the stagnation issue. Thus, dimensional stagnation is a secondary issue and stagnation is a primary issue. In addition, by addressing the stagnation issue the movement of particles is guaranteed, no matter whether the swarm size is small or large. Thus, the swarm size issue (recall that the swarm size issue was defined as particles ceasing to move in the early stage of the optimization process when the swarm size is small) is resolved if stagnation is addressed. Hence, the swarm size issue is categorized as a secondary issue.

Also, the problem scale issue is resolved if the local convergence issue is addressed because the local convergence is independent of the number of dimensions. Thus, if an algorithm is locally convergent it can improve the initial solution for any number of dimensions. Hence, the local convergence is a primary and the problem scale is a secondary issue. Note that

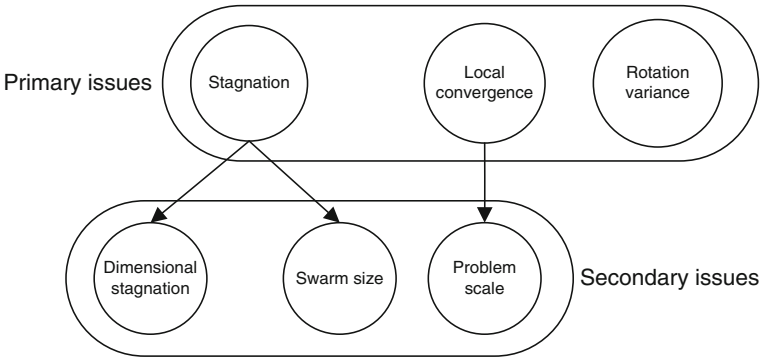

Fig. 2 Diagram of dependencies among PSO issues

the local convergence issue is addressed for an algorithm if the probability of finding the optimality region grows to one. Thus, candidate solutions might stop moving if the best found solution is in the optimality region. However, stagnation is addressed only if the particles are moved in the next steps of the algorithm with non-zero probability (no matter whether they are in the optimality region or not). Also, to address stagnation, there is no need to guarantee improving the solutions in each step and only moving the solutions is enough. Nevertheless, in addressing the local convergence issue, the improvement of the best found solution should happen with non-zero probability. Hence, local convergence and stagnation issues are independent of each other. The diagram in Fig. 2 shows dependencies among the issues.

By addressing stagnation, the swarm size and the dimensional stagnation issues are resolved (Fig. 2). Also, by addressing the local convergence issue, the problem scale issue is resolved. The rotation variance issue is independent of the other discussed issues. Also, the local convergence and the stagnation issues are independent of each other. Thus, the focus of this paper is on addressing three issues: stagnation, local convergence, and rotation variance. In the rest of this paper these three issues are referred to as the *primary* issues.

4 Proposed method

A new general form of the velocity update rule is proposed in this section to address all the primary issues discussed in Sect. 3.7. The new algorithm is called Locally convergent Rotationally invariant PSO (LcRiPSO). It is proven that this variant of PSO with its new form of the velocity update rule addresses all primary issues at the same time under some identified conditions. These conditions serve as a guideline for extracting specific models from the proposed general model. One particular model is derived (and tested experimentally) from this general model.

4.1 Proposed velocity update rule—general form

The proposed general form of the velocity update rule follows the formulation introduced in FIPS (Eq. 7) with some important modifications. Eqs. 12a and 12b show the original velocity update rule in FIPS and the proposed velocity update rule, respectively. The position update

rule remains the same as in SPSO (see Sect. 2.1).

$$\vec{V}^i_{t+1} = \omega \vec{V}^i_t + \sum_{k \in T^i_t} \varphi_k R^i_{kt} \left(\vec{p}^k_t - \vec{x}^i_t \right), \tag{12a}$$

$$\vec{V}^i_{t+1} = \omega \vec{V}^i_t + \sum_{k \in T^i_t} \varphi_k r^i_{kt} \left(f^i_{kt}(\vec{p}^k_t) - \vec{x}^i_t \right). \tag{12b}$$

In Eq. 12b, $f^i_{kt} : R^n \rightarrow R^n$ is a function[6] (stochastic or non-stochastic) which applies perturbations on its inputs. Also, $r^i_{kt} \in R$ is a random scalar (rather than a random diagonal matrix) in the interval [0, 1]. Note that, for each particle i in the swarm we consider $i \in T^i_t$.

The idea behind this new velocity update rule is that the application of the function f^i_{kt} should allow LcRiPSO algorithm to overcome the issues discussed in Sect. 3. In the rest of this section, several conditions for designing this function are provided such that the primary issues (and consequently all secondary issues) are addressed. This function is investigated independently of the iteration number (t), the particle index (i), and the neighbor set (T^i_t). Thus, for the sake of simplicity, the notation $f(.)$ is used in the rest of this section to refer to $f(.) = \left\{ f^i_{kt}(.) : \forall i \forall t \forall k \in T^i_t \right\}$.

The application of function f should allow particles to escape from the stagnation situation. This can be accomplished in various ways. One possibility includes defining a family of functions $f^{i,1}, \ldots, f^{i,m}$ (different from each other) for each particle i and making a random choice of a function $f^{i,m}$ when the velocity update rule is invoked. Another possibility could be based on the function f itself—a stochastic function f would provide the desired characteristic for the velocity update rule (Eq. 10), thus preventing stagnation. Hence, the general condition that should be satisfied by f to overcome the stagnation in the proposed method is

$$\forall t \exists i \exists k \in T^i_t \, P \left(\vec{x}^i_t \neq f \left(\vec{p}^k_t \right) \right) > 0. \tag{13}$$

Indeed, if the function f in Eq. 12b is designed in a way that Eq. 13 is satisfied then LcRiPSO algorithm would not have the stagnation issue.[7]

Another issue that should be addressed in SPSO is the local convergence issue. In Appendix 1, it is proven that if f satisfies the condition given by Eq. 14, LcRiPSO guarantees the convergence condition.

$$\forall \vec{y} \in S \exists A_y \subseteq S \forall \vec{z} \in A_y \; \forall \delta > 0 \quad P \left(|f(\vec{y}) - \vec{z}| < \delta \right) > 0, \tag{14}$$

where A is an open set and S is the search space. In fact, if the function f is designed in such a way that for any input vector $\vec{y}$ in the search space, there exists a region A which contains $\vec{y}$ and $f(\vec{y})$ can be any location within A with non-zero probability, the algorithm addresses the local convergence issue. This condition is called *local convergence condition* throughout the paper.

The rotation variance is also addressed by the proposed method if the function f is designed in a proper way. In Appendix 2, it is proven that the proposed velocity update rule addresses the rotation variance issue if $f(\vec{y})$ satisfies the condition

$$\forall s \in R \forall Orth(Q) \, \forall \vec{b}, \vec{y} \in R^d \, s Q f(\vec{y}) + \vec{b} = \hat{f}(s Q \vec{y} + \vec{b}), \tag{15}$$

[6] Note that we represent this function by a lowercase f. This should not be confused with the uppercase F which represents the objective function.

[7] Note that, in continuous space, it is impossible to hit a point ($\vec{x}^i_t \neq f \left(\vec{p}^k_t \right)$ is always true). However, in the computer simulation, because of the limited floating point precision, it is possible that $\vec{x}^i_t = f \left(\vec{p}^k_t \right)$.

where *Orth(Q)* indicates that the matrix Q is an orthogonal matrix (see Appendix 2 for the proof). In the rest of this paper, hatted ($\widehat{\ }$) variables, vectors, and operators are the ones in the transformed space.

Based on the local convergence condition (Eq. 14), $f(\vec{p}_t^{\,k})$ can be any location in a definable region A. Thus, it has non-zero probability to be different from $\vec{x}_t^{\,i}$, i.e., $P\left(\vec{x}_t^{\,i} \neq f\left(\vec{p}_t^{\,k}\right)\right) > 0$. Hence, if the local convergence condition is satisfied (Eq. 14) by the function f, stagnation overcoming condition (Eq. 13) is satisfied automatically.

To summarize, in order to address all primary issues (and consequently all six issues discussed in Sect. 3), $f(\vec{y})$ needs to satisfy the following conditions:

Condition 1. $\forall \vec{y} \in S\ \exists A_y \subseteq S\ \ \forall \vec{z} \in A_y\ \ \forall \delta > 0\ \ P\left(|f(\vec{y}) - \vec{z}| < \delta\right) > 0$

Condition 2. $\forall s \in R\ \forall\, Orth(Q)\ \ \forall \vec{b},\ \vec{y} \in R^d\ sQf(\vec{y}) + \vec{b} = \hat{f}(sQ\vec{y} + \vec{b})$

Condition 1 addresses stagnation, and consequently the swarm size and the dimensional stagnation issues, as well as local convergence, and consequently the problem scale issue, while the condition 2 addresses the rotation variance issue.

In the next subsection, a particular model of the function f is designed and its ability to address all six issues is examined.

4.2 Proposed velocity update rule—specific model

Any function f which satisfies the two conditions given in Sect. 4.1 guarantees to address stagnation, dimensional stagnation, swarm size, local convergence, problem scale, and rotation variance. There are numerous ways to design such function; in this paper we consider $f_{kt}(\vec{y}) = N(\vec{y},\ \sigma_{kt}^2 I)$, where I is the identity matrix[8] and σ_{kt} is a scalar. The function $N(\vec{y},\ \sigma_{kt}^2 I)$ satisfies both mentioned conditions if $\sigma_{kt} > 0$ for all k and t:

1) if $\sigma_{kt} > 0$ for all k and t, $N(\vec{y},\ \sigma_{kt}^2 I)$ satisfies condition 1 because it can sample an arbitrarily location within a hyper-sphere with center y and non-zero radius (open set A). Thus, for any y at any t, a region A (that contains y) exists whereby $N(\vec{y},\ \sigma_{kt}^2 I)$ can sample any location in that region with non-zero probability.
2) if $\sigma_{kt} > 0$ for all k and t, $N(\vec{y},\ \sigma_{kt}^2 I)$ satisfies condition 2 (see Appendix 3 for a proof).

For this specific model, we consider the global best topology for the swarm, as it is simple and frequently used in other PSO research papers (Clerc and Kennedy 2002) (note that it does not mean that the global best topology is the best topology to be used for this algorithm). Also, the parameters ω, φ_1, and φ_2 are fixed to 0.7298, 1.4962, and 1.4962, as these values were defined in Clerc and Kennedy (2002) and they are widely used in many PSO variants. Note that setting the parameters of the algorithm is not the aim of this study as the paper focuses on the ability of the algorithm to address the issues introduced in Sect. 3. However, the parameters of the proposed algorithm are set in Sect. 5.2 through a simple procedure. Also, one can consider advanced parameter setting tools (e.g., Hutter et al. 2010).

Equation 16 shows the velocity update rule for the proposed specific model

$$\vec{V}_{t+1}^{\,i} = \omega \vec{V}_t^{\,i} + \varphi_1 r_{1t}^i \left(\underbrace{N\left(\vec{p}_t^{\,i}, \sigma_{1t}^2 I\right)}_{f_{1t}\left(\vec{p}_t^{\,i}\right)} - \vec{x}_t^{\,i} \right) + \varphi_2 r_{2t}^i \left(\underbrace{N\left(\vec{g}_t, \sigma_{2t}^2 I\right)}_{f_{2t}\left(\vec{g}_t\right)} - \vec{x}_t^{\,i} \right), \tag{16}$$

[8] Note that the function f is the same for all particles; however, it is different for different neighbors (variance is different for different neighbors). Thus, we refer to these functions as $f_{kt}(y)$ because they are independent of i.

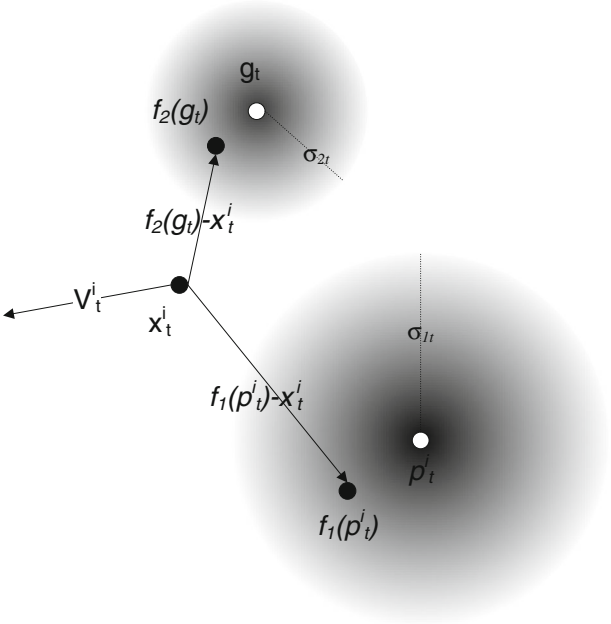

Fig. 3 The function $N(y, \sigma_{kt}^2 I)$ generates two new locations around g and p, $f(g) = N(g, \sigma_{1t}^2 I)$, $f(p) = N(p, \sigma_{2t}^2 I)$ The average of the distance between new locations ($f(p)$ and $f(g)$) and the original locations (p and g) is related to the variance of the normal distribution. Note that the figure is not completely accurate since a Normal distribution does not have a compact support

where σ_{1t}^2 and σ_{2t}^2 are the variances of the normal distributions ($N\left(\vec{p}_t^{\,i}, \sigma_{1t}^2 I\right)$ and $N\left(\vec{g}_t, \sigma_{2t}^2 I\right)$. The value of the variance plays a key role in this formulation and it is investigated in the rest of this subsection. Figure 3 shows how this algorithm works.

The value of variance σ_{kt}^2 affects the area in which $f_{1t}\left(\vec{p}_t^{\,i}\right)$ and $f_{2t}\left(\vec{g}_t\right)$ might be in. In fact, a big value for the variance results in a higher probability of generating $f_{1t}\left(\vec{p}_t^{\,i}\right)$ far from $\vec{p}_t^{\,i}$ or $f_{2t}\left(\vec{g}_t\right)$ far from $\vec{g}_t$. Also, a small value for the variance results in a higher probability for generating $f_{1t}\left(\vec{p}_t^{\,i}\right)$ close to $\vec{p}_t^{\,i}$ or $f_{2t}\left(\vec{g}_t\right)$ close to $\vec{g}_t$. Note that there are some similarities between this operator and the parent-centric crossover (PCX) (Deb et al. 2002).

Exploring the search space in the earlier stage of the optimization process and exploiting solutions around quality known solutions in the later stages of the optimization process is normally preferable in an optimization algorithm. On the other hand, according to Eq. 16, in each iteration t new random locations around $\vec{p}_t^{\,i}$ and $\vec{g}_t$ (i.e., $f_{1t}\left(\vec{p}_t^{\,i}\right)$ and $f_{2t}\left(\vec{g}_t\right)$ are generated by a normal distribution N with variance σ_{kt}^2. In fact, the average distance between $\vec{p}_t^{\,i}$ and $f_{1t}\left(\vec{p}_t^{\,i}\right)$ and $\vec{g}_t$ and $f_{2t}\left(\vec{g}_t\right)$ is controlled by the variance of the distribution (Fig. 3). Hence, in order to perform exploration in the earlier stages and exploitation in the later stages, larger values of variance at the beginning of the optimization and smaller values at the later stages are preferable. Note that the explorative and exploitative behavior of the particles depends also on the values of ω, ϕ_1, and ϕ_2. However, these values are considered constant in our model so that their influence on the behavior of the particles is not changed during the run. To determine the value of the variance, we can consider that it is a function of some other variables, i.e., $\sigma_{kt} = h_{kt}(.)$ for $k = 1, 2$, where h_{kt} is a function of some variables. In this case, $N(y, h_{kt}^2(.)I)$ satisfies conditions 1 and 2 if $h_{kt} > 0$ for all k and t and $h_{kt} : R^d \rightarrow R$,

i.e., h_{kt} returns a scalar (see Appendix 3 for a proof). Thus, two main characteristics of the function h_{kt} to determine σ_{kt} are summarized as:

(1) $h_{kt} > 0$ and $h_{kt} : R^d \rightarrow R$ to satisfy conditions 1 and 2, respectively, and
(2) larger values in the exploration phase (earlier stage of the optimization) and smaller values in the exploitation phase (later stage of the optimization) are preferable.

There might be many different ways to design the function h_{kt}, and each may have a different impact on the performance of the algorithm. In this paper, we propose to use a function based on the Euclidean distance between $\vec{x}_t^i$ and $\vec{p}_t^i$ or $\vec{x}_t^i$ and $\vec{g}_t$ for h_{1t} and h_{2t}. Indeed, the function h_{kt} is defined as ($h_{1t} = h_{2t} = h_t$):

$$D_t = h_t(\vec{y}, \vec{z}) = \begin{cases} l D_{t-1} & \text{if } y^i = z^i \text{ for all } i \\ l\sqrt{\sum_{i=1}^{d}(y^i - z^i)^2} & \text{otherwise,} \end{cases} \tag{17}$$

where y and z are two input vectors and l>0 is a constant. Note that whenever the two inputs of the function $h_t(\vec{y}, \vec{z})$ are equal, the previously calculated distance is used instead. This guarantees $h_{kt} > 0$ for all t. Also, the value of l has an important impact on the performance of the algorithm and it is set by experiments in Sect. 5. The velocity update rule in Eq. 16 is revised to the following form:

$$\vec{V}_{t+1}^i = \omega\vec{V}_t^i + \varphi_1 r_t^i \left(N\left(\vec{p}_t^i, h_t^2\left(\vec{x}_t^i, \vec{p}_t^i\right) I\right) - \vec{x}_t^i\right) + \varphi_2 r_t^i \left(N\left(\vec{g}_t, h_t^2\left(\vec{x}_t^i, \vec{g}_t\right) I\right) - \vec{x}_t^i\right). \tag{18}$$

In all of the further tests in this section, LcRiPSO ($\sigma = *$) refers to the formulation in Eq. 16 (where * is the value for σ that is specified in each test) and LcRiPSO ($l = *$)[9] refers to the formulation in Eq. 18 (where * is the value for l that is specified in each test).

4.3 Experimental validation

In all of the following tests, the number of function evaluations is set to $5000d/n$ (n is the number of particles) and all results are the averages over 50 independent runs. Also, the value of l in the next experiments in this subsection have been selected by some trials (see Sect. 5.2 for a more comprehensive discussion about setting these parameters). However, the value of σ has remained unchanged in all test functions in this subsection to show the ability of the method to address the issues in Sect. 3.

In each test, the goal is to assess the ability of the proposed method to maintain its performance in different situations (rotating search space, reducing the swarm size, and increasing the number of dimensions). It is not our goal to compare the performance of different methods. Also, note that the ability of the method for addressing these issues has been proven and it is not dependent on the parameters l and σ. However, its performance might be different when these parameters change. As an example, the algorithm might perform poorly with a specific value of l or σ and perform better using another value, but in both cases, its performance is not changed by rotating the search space.

In order to show the ability of the proposed method in dealing with the swarm size issue, the method is applied to the first function in the CEC08 benchmark (Tang et al. 2007) (that is a shifted sphere function), with $n \in \{2, 3, 4, 5, 6, \ldots, 20\}$ and $d = 10$. The average of the performance ratio, defined as performance ratio $= \frac{\text{objective value of the final solution}}{\text{objective value of the initial solution}}$, is reported

[9] A Matlab source code for this variant is available as online supplementary material.

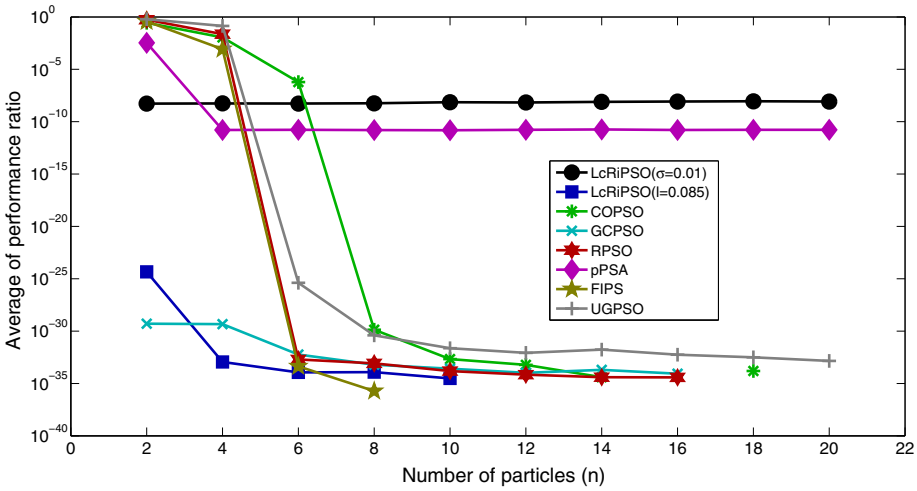

Fig. 4 The *horizontal* axis is n and the vertical axis is the average of the performance ratio over 50 runs. Note that for the cases where the average of performance ratio is zero, the value has not been shown in the graph as the *vertical* axis is in the logarithmic scale (Color figure online)

for each case. The value of this ratio is one if no improvement takes place during the run. Results are shown in Fig. 4.

In this test, the value of l is equal to 0.085 and the value of σ is 0.01. Figure 4 demonstrates that the performance of COPSO, pPSA, FIPS, and RPSO is dramatically impaired when the swarm size (n) is small. Also, results show that the performance ratio is around 1 for COPSO, FIPS, UGPSO, and RPSO when $n = 2$. This means that for small n, these algorithms could not even improve the initial solution and that they stop improving at the very early stage of the optimization process. Thus, this test serves as a counter example to show that COPSO, FIPS, UGPSO, and RPSO cannot address the swarm size issue in the general case (see also Sect. 5.5 for a more comprehensive experiment on this issue). On the other hand, it is obvious that the performance ratio of GCPSO and LcRiPSO ($l = 0.085$) is much smaller than the other variants when n is small. Hence, we can say that these two methods have effectively addressed the swarm size issue. Note that this test only shows that the proposed method is not stagnated even with a small number of particles. However, it is obvious that the performance of the method depends on the values of the parameters l and σ, which is also the case for other methods.

In order to study the performance of LcRiPSO in terms of addressing the problem scale issue, the method is applied to the first function in the CEC08 benchmark (CEC08_1, the first function in the benchmark CEC08 that is a shifted sphere function) with different number of dimensions ($d = \{10, 60, \ldots, 510\}$). In Fig. 5, the performance ratio is reported.

In this test, the value of n was set equal to d (number of dimensions) for all algorithms and each algorithm was applied to the CEC08_1 test function 50 times. The performance ratio was used, rather than the original performance measure, to show the ability of the methods to improve initial solutions (note that it is expected that the algorithms improve the initial solutions even if the number of dimensions is very large). Figure 5 indicates that the performance ratio for GCPSO, COPSO, RPSO, and FIPS deteriorates with the growth of the number of dimensions. The average of the performance ratio is very close to 1 (larger than 0.999) on average (over 50 runs) for these algorithms when d is larger than 110, 260, 260,

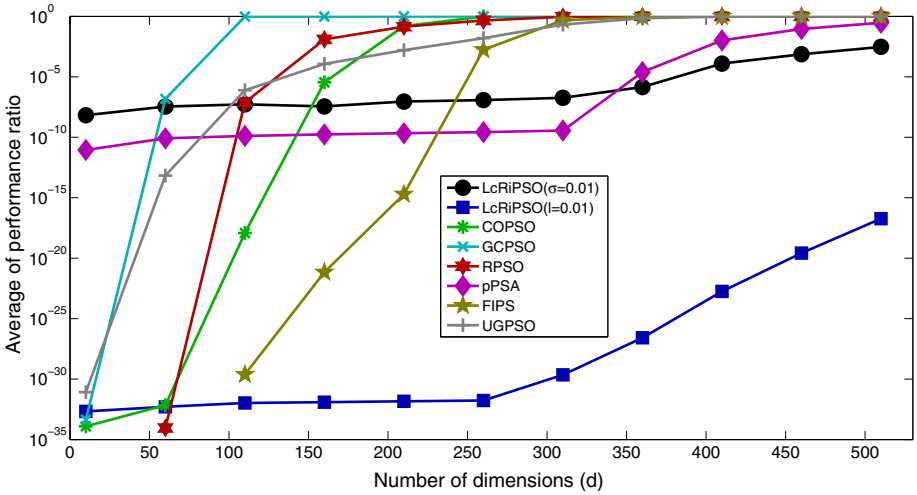

Fig. 5 The *horizontal* axis is the number of dimensions and the vertical axis is the average performance ratio over 50 runs. The *horizontal line* indicates different values of the performance ratio, starting from 1.0. Note that for the cases where the average of performance ratio is zero, the value has not been shown in the graph as the *vertical* axis is in the logarithmic scale (Color figure online)

and 310, respectively. For the UGPSO, the impairment happened later, but the performance ratio of this algorithm is close to one when $d > 360$. Thus, it is obvious that there is no guarantee that these methods improve the initial solutions when the number of dimensions grows. For pPSA, the performance ratio becomes larger than 0.95 when $d = 510$. The proposed algorithm (LcRiPSO ($\sigma = 0.01$) and LcRiPSO ($l = 0.01$)) could improve the initial solution in all scales of the problem. This test shows that LcRiPSO is able to improve the initial solution even when the number of dimensions grows. Note that it does not mean that there is no other choice for the algorithm's parameter (l and σ) which works better when the number of dimensions grows. In addition, the performance of LcRiPSO becomes slightly worse when the number of dimensions increases; however, it still offers improvement. One potential reason behind this drop in the performance is that the number of function evaluations is not large enough for higher dimensional test functions to achieve the same performance. Also, another potential reason is that the setting of the parameters of the method was not optimal (our simulation with $\sigma = 0.007$ for LcRiPSO gives the average of 1.82E−24 for 510 dimensional functions). In fact, different choices for l or σ affect the performance of the method and more robust behavior (in terms of performance in higher dimensions) might appear by choosing different values for these parameters in different situations (e.g., number of dimensions, number of particles). Note that dependency among parameters might exist in any other optimization method.

In order to examine the rotation variance issue, LcRiPSO is applied to a test function called the Ellipse function (Spears et al. 2010). This test function was specifically designed to test the ability of optimization methods in dealing with the rotation of the search space. Each algorithm was applied to this test function 50 times and was run for 500 function evaluations. Then function was rotated by 5° and the algorithms were applied to the rotated function. This process was repeated until the rotation achieved 180°. Figure 6 shows the average performance (the quality of the found solutions) over 50 runs for each algorithm with different rotations of the search space.

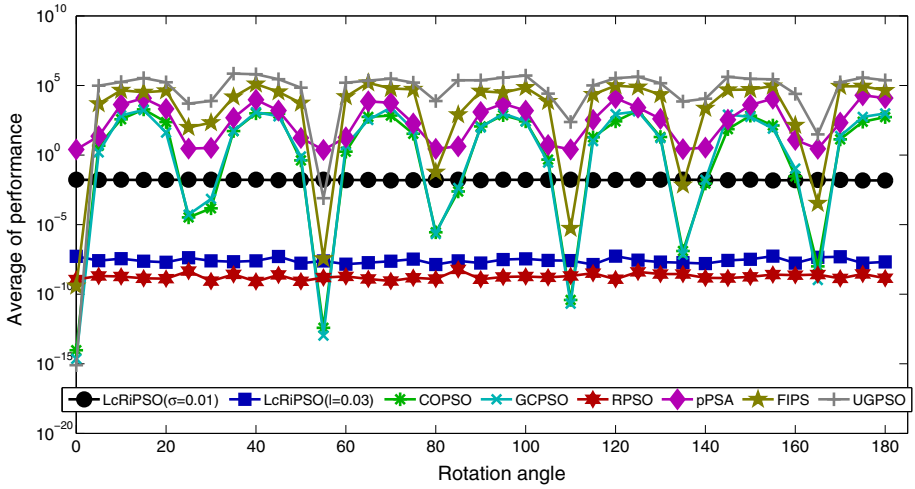

Fig. 6 The *horizontal* axis is the rotation angle and the *vertical* axis is the average performance of the algorithm (over 50 runs) (Color figure online)

Table 1 Addressing various issues by different variants of pso algorithm

	Stagnation	Dimensional stagnation	Swarm size	Rotation variance	Problem scale	Local convergence
COPSO						
RPSO		Addressed		Addressed		
GCPSO	Addressed	Addressed	Addressed			Addressed
UGPSO					Addressed	
FIPS						
pPSA	Addressed	Addressed	Addressed		Addressed	Addressed
LcRiPSO	Addressed	Addressed	Addressed	Addressed	Addressed	Addressed

In this experiment, the number of dimensions was set to 2 and the number of particles for all algorithms was set to 20.

In this experiment, LcRiPSO ($\sigma = 0.01$) (Eq. 16), and LcRiPSO ($l = 0.03$) (Eq. 18) were used. This figure indicates that pPSA, FIPS, GCPSO, COPSO, and UGPSO are sensitive to the rotation of the search space (their performance is changed by rotating the search space). Thus, this example shows that these methods are rotationally variant. However, the performance of RPSO, LcRiPSO ($\sigma = 0.01$), and LcRiPSO ($l = 0.03$) is not changed when the search space is rotated.

Table 1 summarizes the abilities of the investigated methods in addressing all issues introduced in Sect. 3.

Table 1 indicates that the only algorithm which has addressed all issues at the same time is the proposed algorithm (LcRiPSO).

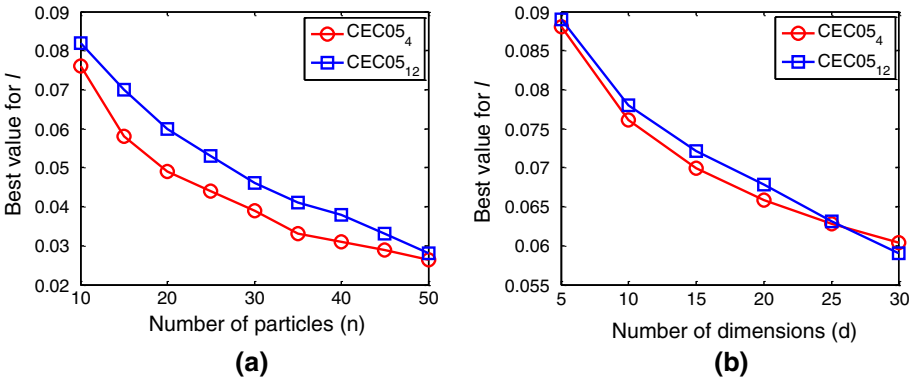

Fig. 7 Parameter setting of the proposed algorithm when n and d are changed. Best value of l versus number of particles (**a**) and versus number of dimensions (**b**). The number of dimensions was set to 10 in (**a**) and the number of particles was set to 10 in (**b**) (Color figure online)

5 Experiments and comparisons

In this section, we define the test bed and we analyze the parameter l of the proposed method. Then, we apply several PSO variants to some benchmark test functions and compare their results. In all further comparisons, we use LcRiPSO with Eq. 18 as velocity update rule.

5.1 Experimental setup

All codes were implemented under the MatlabR2011a environment. Due to the stochastic nature of the methods, the reported results for each algorithm represent the averages over 50 runs. The maximum number of function evaluations (FE) is fixed to $5000d/n$ in all tests. In order to evaluate the algorithm performance, the methods are applied to the benchmark functions CEC05 (the first 14 test functions) (Suganthan et al. 2005) and CEC08 (Tang et al. 2007). The tests are done for different number of dimensions (varied from 2 to 1000) and swarm sizes (varied from 2 to 1000). Wilcoxon test has been used to compare the difference between the results of LcRiPSO and other methods (all p's < 0.05).

5.2 Parameter setting

The only parameter that was added to SPSO velocity update rule was l which is the coefficient of the distance function. Although an adaptive approach for controlling this parameter might be preferable, in this paper we set the parameter manually. The experiment is done for one unimodal and one multimodal function from CEC05 (Suganthan et al. 2005) benchmark ($CEC05_4$ and $CEC05_{12}$). The method is applied to these functions with various values of n and d.

Figure 7 shows that the best value for l is dependent on both n and d. The pattern of this dependency is very similar for all other CEC benchmarks, but the baseline for different functions was slightly different. Our experiments showed that the following formulation well-matches with this pattern:

$$l = \frac{a_1}{n^{a_2}} \frac{a_3}{d^{a_4}}.$$

The values for a_1, a_2, a_3, and a_4 were explored by experiments. Results showed that in most cases the following values work better than other values over all tested functions and parameters values: $(a_1, a_2, a_3, a_4) = (0.91, 0.21, 0.51, 0.58)$. Hence, these values are used throughout the paper for identification of l. Note that this is not a comprehensive study on the introduced parameter and obviously an adaptive parameter is more reliable when different optimization problems are at hand. However, this is not investigated in this paper and it is left for future study.

5.3 Comparison

In order to evaluate the performance of the LcRiPSO, the results of four tests are reported and analyzed.

1. In the first test, several algorithms are applied to CEC05 benchmark when d is set to 2, 10, 30, and 50 and $d = n$. This test is performed to show the overall ability of LcRiPSO in optimizing benchmark functions in comparison to other variants. In this test, COPSO and FIPS (with ring topology) are used for comparison purposes. These have been selected because they are among the most-studied PSO-based methods and can be counted as baseline methods for comparison. The results of RPSO, pPSA, UGPSO, and GCPSO are also reported and compared with the results of LcRiPSO.
2. In the second test, the algorithms are applied to the same benchmark functions (CEC05), but the number of particles n for all methods is fixed to 2. This test is responsible for measuring the performance of the methods when the number of particles is small in comparison to the number of dimensions (d is set to 10 and 30 for this test). In this test, LcRiPSO is compared with COPSO in order to demonstrate the ability of the proposed method with small swarm size in comparison to a well-studied PSO variant (i.e., COPSO). Also, LcRiPSO is compared with pPSA and GCPSO in this situation because, as outlined in Sect. 3, these methods have addressed the swarm size issue as well.
3. In the third test, LcRiPSO, FIPS, COPSO, RPSO, and UGPSO are compared with each other in 100 dimensional spaces when CEC05 is used as the benchmark. Also, LcRiPSO, COPSO, and UGPSO are applied to higher dimensional benchmark functions (CEC08) with $d = \{500, 1000\}$. The aim of this test is to show that the proposed algorithm can converge to better solutions even in large scale test functions, while this is not the case for the other listed methods.
4. In the fourth test, LcRiPSO, RPSO, and COPSO are applied to some standard test functions and their rotated versions. The aim of this test is to evaluate if the performance of LcRiPSO is changed by rotating the search space.

5.4 Test 1, overall performance

The performance of the proposed LcRiPSO method is compared with COPSO, FIPS, and RPSO. Tables 2 and 3 show the results.

For each test, if a result of LcRiPSO in a test is significantly better than some other methods, then that result has been superscripted by the initials of the names of the other algorithms. For example, superscripting a value by CF in the LcRiPSO column, e.g., $\#\#\#\#^{CF}$, means that LcRiPSO performs significantly (according to the Wilcoxon test, $p < 0.05$) better than COPSO and FIPS in that test. Also, if any other method performs significantly better than LcRiPSO on a test function, its result has been superscripted by L. Note that we only compare LcRiPSO with other methods; comparing other methods with each other was not the aim of this test. Results show that LcRiPSO has significantly better performance than COPSO

Table 2 Comparison of the overall performance of LcRiPSO when it is applied to CEC05 (number of dimensions was $d = 2$ and $d = 10$ and number of particles was set equal to number of dimensions, i.e., $d = n$)

Function	$d=2$				$d=10$			
	COPSO	FIPS	RPSO	LcRiPSO	COPSO	FIPS	RPSO	LcRiPSO
$CEC05_1$	3.39E+01	1.24E+02	8.01E+01	**0.00E+00**^{CFR}	3.40E−28	2.44E−02	1.35E−28	**0.00E+00**^{CR}
$CEC05_2$	7.95E+01	2.51E+02	1.12E+03	**0.00E+00**^{CFR}	7.72E−27	1.15E+02	3.48E−28	**1.63E−28**^{CFR}
$CEC05_3$	4.86E+03	4.93E+03	3.37E+08	**2.50E+03**^{CFR}	8.07E+04	5.30E+05	**1.32E+04**^{L}	5.17E+04^{F}
$CEC05_4$	2.15E+02	2.26E+02	9.94E+02	**0.00E+00**^{CFR}	4.76E+02	1.42E+03	2.51E−15	**1.62E−28**^{CFR}
$CEC05_5$	1.12E+03	9.15E+02	1.97E+03	**1.16E+02**^{CFR}	1.77E+02	1.10E+03	**1.03E+00**	1.93E+01^{CFR}
$CEC05_6$	5.00E+01	3.70E+07	7.92E+01	**2.46E+01**^{CFR}	**6.28E+00**^{L}	6.00E+02	4.56E+01^{L}	2.97E+02^{F}
$CEC05_7$	3.65E+00	9.94E−01	2.10E+00	**2.60E−02**^{CFR}	7.71E−01	7.11E−01	9.37E−01	**3.38E−01**^{CR}
$CEC05_8$	1.94E+01	1.97E+01	1.83E+01	**1.19E+01**^{CF}	2.03E+01	2.01E+01	2.04E+01	**2.03E+01**^{F}
$CEC05_9$	3.92E+00	2.36E+00	4.83E+00	**8.95E−01**^{CFR}	1.63E+01^{L}	**4.32E+00**^{L}	3.79E+01	2.39E+01^{R}
$CEC05_{10}$	3.43E+00	4.25E+00	9.18E+00	**1.02E+00**^{CFR}	2.72E+01	**1.65E+01**^{L}	4.25E+01	2.67E+01^{CR}
$CEC05_{11}$	4.90E−01	3.77E−01	9.82E−01	**5.49E−02**^{CFR}	6.18E+00	**4.34E+00**	7.81E+00	4.76E+00^{CR}
$CEC05_{12}$	1.39E+02	9.71E+02	1.03E+03	**1.55E+01**^{CFR}	**4.11E+03**	7.98E+02	5.40E+03	4.43E+03
$CEC05_{13}$	5.45E+00	1.04E−01	5.67E+01	**1.64E−02**^{CFR}	1.43E+00	**1.40E+00**^{L}	3.45E+00	1.44E+00^{R}
$CEC05_{14}$	5.79E−01	5.70E−01	6.66E−01	**4.71E−02**^{CFR}	3.41E+00	3.38E+00	3.57E+00	**3.20E+00**^{CR}

The best results have been bolded in each instance

and FIPS in all 14 cases when $d = 2$. Compared to RPSO, in 13 cases (of 14) the proposed method performs significantly better, while only in one case ($CEC05_8$) these two methods perform statistically the same. For $d = 10$, LcRiPSO has significantly better performance in 8, 6, and 10 functions when compared to COPSO, FIPS, and RPSO, respectively. On the other hand, these methods perform significantly better than LcRiPSO in 2, 3, and 2 cases, respectively.

In the 30 dimensional cases (Table 3), LcRiPSO performs significantly better than COPSO, FIPS, and RPSO in 6, 4, and 3 functions while it performs significantly worse than these methods in 4, 3, and 0 functions, respectively. Finally, when $d = 50$, in 5, 4, and 3 functions the method reached significantly better solutions than COPSO, FIPS, and RPSO, respectively. However, these methods perform significantly better than LcRiPSO in 4, 4, and 0 functions, respectively. These comparisons show that the algorithm offers significant improvement (with reference to well-studied PSO methods) when a small size test function is at hand ($d = 2$, $d = 10$). However, in the middle size test functions ($d = 30$, $d = 50$), the proposed method performs better than the previous methods in only a limited number of test functions and performs the same in most cases.

Table 4 shows the results of comparisons among LcRiPSO, pPSA, GCPSO, and UGPSO. Results show that LcRiPSO performs significantly better than pPSA, GCPSO, and UGPSO in 7, 6, and 10 cases, respectively, over all 14 cases for 10 dimensional functions. LcRiPSO performs better than those methods in 6, 6, and 7 cases in the 30 dimensional functions.

5.5 Test 2, small swarm size

In this test, the performance of the proposed method is tested when the swarm size is small ($n = 2$). Table 5 shows the results.

Table 3 Comparison of the overall performance of LcRiPSO when it is applied to CEC05 (number of dimensions was $d = 30$ and $d = 50$ and number of particles was set equal to number of dimensions, i.e., $d = n$)

Function	$d=30$				$d=50$			
	COPSO	FIPS	RPSO	LcRiPSO	COPSO	FIPS	RPSO	LcRiPSO
$CEC05_1$	2.59E−27	**0.00E+00**	1.01E−29	1.97E−28^{C}	4.55E−26	**0.00E+00**	**0.00E+00**	7.49E−28
$CEC05_2$	1.14E−08	9.38E−01	4.69E−18	**1.09E−24**^{CFR}	8.51E−01	5.27E+02	2.17E−04	**8.05E−15**^{CFR}
$CEC05_3$	9.45E+05	4.12E+06	2.47E+05	**2.43E+05**^{CF}	2.95E+06	9.97E+06	1.16E+06	**2.60E+05**^{CF}
$CEC05_4$	2.52E+03	6.57E+03	9.72E+00	**6.33E+00**^{CFR}	1.79E+04	2.15E+04	7.55E+03	**2.94E+03**^{CFR}
$CEC05_5$	5.64E+03	4.30E+03	**4.27E+03**	4.63E+03^{C}	1.23E+04	8.64E+03	8.80E+03	**8.51E+03**^{C}
$CEC05_6$	**3.38E+01**^{L}	1.16E+02	2.92E+02	5.25E+02	**7.92E+01**^{L}	1.59E+02	4.19E+02	6.41E+02
$CEC05_7$	**1.75E−02**	1.94E−01	1.81E−02	1.79E−02	1.24E−02	**8.83E−03**	1.27E−02	1.09E−02^{R}
$CEC05_8$	2.09E+01	2.07E+01	2.09E+01	**2.09E+01**^{CFR}	2.11E+01	2.12E+01	2.11E+01	**2.10E+01**^{CF}
$CEC05_9$	9.86E+01^{L}	**4.93E+01**^{L}	1.72E+02	1.65E+02	2.01E+02^{L}	**1.47E+02**^{L}	3.34E+02	3.56E+02
$CEC05_{10}$	**1.33E+02**^{L}	1.73E+02^{L}	2.20E+02	2.20E+02	**2.99E+02**^{L}	4.32E+02^{L}	4.91E+02	5.58E+02
$CEC05_{11}$	2.84E+01	2.77E+01	2.85E+01	**2.82E+01**	**5.52E+01**	5.95E+01	5.54E+01	5.72E+01
$CEC05_{12}$	1.40E+04	**1.39E+04**	2.42E+04	3.10E+04	**6.28E+04**	7.33E+04	1.40E+05	1.15E+05
$CEC05_{13}$	**6.99E+00**^{L}	1.06E+01^{L}	1.29E+01	1.36E+01	**1.58E+01**^{L}	2.47E+01^{L}	2.55E+01	3.55E+01
$CEC05_{14}$	1.27E+01	**1.27E+01**	1.28E+01	1.29E+01	2.22E+01	2.25E+01^{L}	**2.22E+01**	2.23E+01

The best results have been bolded in each instance

Table 4 Comparison of the overall performance of LcRiPSO when it is applied to CEC05 (number of dimensions was $d = 10$ and $d = 30$ and number of particles was set equal to number of dimensions, i.e., $d = n$)

Function	$d=10$				$d=30$			
	pPSA	GCPSO	UGPSO	LcRiPSO	pPSA	GCPSO	UGPSO	LcRiPSO
$CEC05_1$	8.68E−30	7.57E−30	8.12E−27	**0.00E+00**^{pGU}	**5.26E−29**	6.35E−29	1.16E−27	1.97E−28^{U}
$CEC05_2$	8.36E−28	1.20E−26	2.23E−13	**1.63E−28**^{pGU}	5.68E−08	2.32E−07	2.09E+02	**1.09E−24**^{pGU}
$CEC05_3$	6.77E+04	5.56E+04	1.27E+05	**5.17E+04**^{pU}	8.26E+05	1.02E+06	3.55E+06	**2.43E+05**^{pGU}
$CEC05_4$	5.06E+02	4.29E+02	1.42E−02	**1.62E−28**^{pGU}	4.24E+03	3.32E+03	1.30E+04	**6.33E+00**^{pGU}
$CEC05_5$	**1.11E+00**^{L}	2.03E+01	7.06E+02	1.93E+01^{U}	6.54E+03	6.58E+03	6.98E+03	**4.63E+03**^{pGU}
$CEC05_6$	9.31E+00	**3.95E+00**^{L}	9.93E+01	2.97E+02^{U}	**2.91E+01**^{L}	3.57E+01^{L}	1.34E+02	5.25E+02
$CEC05_7$	6.60E−01	6.86E−01	8.19E−01	**3.38E−01**^{pGU}	1.64E−02	2.04E−02	**1.34E−02**	1.79E−02
$CEC05_8$	2.04E+01	2.04E+01	2.04E+01	**2.03E+01**^{pGU}	2.10E+01	2.10E+01	2.10E+01	**2.09E+01**^{pGU}
$CEC05_9$	**1.70E+01**	1.73E+01	9.37E+00	2.39E+01^{U}	1.02E+02^{L}	1.05E+02	**4.81E+01**^{L}	1.65E+02
$CEC05_{10}$	2.66E+01	2.39E+01	**2.14E+01**	2.67E+01	1.99E+02	2.40E+02	**1.05E+02**^{L}	2.20E+02
$CEC05_{11}$	6.57E+00	6.74E+00	6.91E+00	**4.76E+00**^{pGU}	3.21E+01	3.10E+01	3.14E+01	**2.82E+01**^{pGU}
$CEC05_{12}$	3.13E+03	2.36E+03	**6.17E+02**^{L}	4.43E+03	1.62E+04	2.53E+04	**1.28E+04**^{L}	3.10E+04
$CEC05_{13}$	1.10E+00^{L}	1.25E+00	**1.01E+00**^{L}	1.44E+00	9.61E+00	8.35E+00	**7.60E+00**^{L}	1.36E+01
$CEC05_{14}$	**3.01E+00**	3.50E+00	3.20E+00	3.20E+00	1.29E+01	1.29E+01	**1.28E+01**	1.29E+01

The best solution found by the methods has been shown in bold

Table 5 Comparison of the performance of LcRiPSO when it is applied to CEC05 (number of dimensions was $d = 10$ and $d = 30$ and number of particles was set to 2, i.e., $n = 2$)

Function	$d=10$				$d=30$			
	COPSO	GCPSO	pPSA	LcRiPSO	COPSO	GCPSO	pPSA	LcRiPSO
$CEC05_1$	1.02E+04	4.37E−25	4.44E−07	**3.75E−29**^{CGp}	8.72E+04	2.51E+03	6.38E−06	**1.32E−27**^{CGp}
$CEC05_2$	2.01E+04	2.40E−24	5.20E−07	**8.08E−28**^{CGp}	1.57E+05	2.00E+04	1.77E−05	**1.81E−17**^{CGp}
$CEC05_3$	2.04E+08	2.20E+06	**1.32E+05**	1.82E+05^{C}	2.09E+09	4.39E+08	6.80E+05	**5.42E+05**^{C}
$CEC05_4$	2.32E+04	2.79E+04	2.54E+04	**1.25E−13**^{CGp}	2.33E+05	4.04E+05	1.83E+05	**1.20E+03**^{CGp}
$CEC05_5$	1.64E+04	1.32E+04	3.04E+03	**6.65E+02**^{CGp}	4.55E+04	3.74E+04	1.06E+04^{P}	**1.25E+04**^{CG}
$CEC05_6$	9.33E+09	3.06E+02	7.68E+02	**2.22E+02**^{CG}	7.99E+10	7.26E+09	7.16E+02	**7.13E+02**^{C}
$CEC05_7$	4.89E+02	3.28E+00	4.53E+01	**5.53E−01**^{CGp}	3.77E+03	5.89E+02	1.75E−02	**2.28E−02**^{CG}
$CEC05_8$	2.03E+01	2.04E+01	**2.01E+01**^{L}	2.04E+01	2.09E+01	2.08E+01	**2.02E+01**^{L}	2.10E+01
$CEC05_9$	8.14E+01	6.69E+01	6.06E+01	**3.01E+01**^{CGp}	4.51E+02	3.41E+02	3.51E+02	**2.18E+02**^{CGp}
$CEC05_{10}$	1.40E+02	1.23E+02	1.10E+02	**3.48E+01**^{CGp}	8.56E+02	7.52E+02	6.67E+02	**2.87E+02**^{CGp}
$CEC05_{11}$	1.03E+01	1.09E+01	5.57E+00	**5.69E+00**^{CG}	4.08E+01	4.32E+01	3.24E+01	**2.99E+01**^{CGp}
$CEC05_{12}$	1.08E+05	1.98E+04	9.93E+03	**8.26E+03**^{C}	1.49E+06	3.75E+05	7.71E+04	**8.88E+04**^{CG}
$CEC05_{13}$	2.61E+04	1.53E+01	**1.07E+00**^{L}	2.42E+00^{CG}	1.01E+06	1.01E+05	5.12E+00^{L}	**1.95E+01**^{CG}
$CEC05_{14}$	4.29E+00	4.36E+00	4.38E+00	**3.50E+00**^{CGp}	1.40E+01	1.40E+01	1.41E+01	**1.32E+01**^{CGp}

The best solution found by the methods has been shown in bold

In 13 out of 14 cases, LcRiPSO has significantly better performance in comparison to COPSO when the number of dimensions is either 10 or 30. In $CEC05_8$, these two methods have no significant difference. In comparison to GCPSO (Van den Bergh and Engelbrecht 2010), LcRiPSO has better performance in all cases except in $CEC05_8$ in both $d = 10$ and $d = 30$ cases. Finally, for $d = 10$, pPSA performed significantly better than LcRiPSO in only 2 cases. On the other hand, LcRiPSO was significantly better than pPSA in 8.

In the 30 dimensional functions, LcRiPSO shows a significantly better performance in comparison to pPSA in 7 functions and a significantly worse performance in 3 functions. The result of FIPS has not been reported here because its performance is almost the same as COPSO when n is small. These results indicate that the proposed method addresses the swarm size issue.

5.6 Test 3, large scale problems

In order to show the performance of the proposed algorithm in higher number of dimensions, it is applied to 100, 500, and 1000 dimensional test functions. In the first test, LcRiPSO is compared with four PSO variants when they are applied to 100 dimensional test functions taken from CEC05 benchmark. COPSO and FISP were selected for comparison because these are well-studied methods. UGPSO has been selected because it has addressed the same issue (i.e., problem scale).

According to Table 6, LcRiPSO has significantly better performance in comparison to COPSO, FIPS, RPSO, and UGPSO in 8, 9, 10, and 9 functions, respectively. These methods have significantly better performance in only 3, 3, 4, and 3 functions, respectively. In the remaining functions all methods performed statistically the same.

Table 6 Comparison of different methods when $d = 100$ (Cec05 Benchmark)

Function	$d = 100$				
	COPSO	FIPS	RPSO	UGPSO	LcRiPSO
$CEC05_1$	7.06E−12	6.79E−27	1.40E−05	8.39E−02	**3.33E−27**^{CFUR}
$CEC05_2$	6.86E+03	3.16E+04	1.96E+04	1.03E+05	**2.23E−05**^{CFUR}
$CEC05_3$	4.42E+09	5.71E+07	4.16E+07	9.59E+07	**1.71E+06**^{CFUR}
$CEC05_4$	1.16E+05	1.12E+05	1.65E+05	3.08E+05	**5.52E+04**^{CFUR}
$CEC05_5$	3.34E+04	2.34E+04	3.81E+04	4.33E+04	**2.10E+04**^{CFUR}
$CEC05_6$	1.64E+02	1.90E+02	1.69E+02	1.53E+04	**1.13E+02**^{CFUR}
$CEC05_7$	8.03E+02	1.92E−01	7.85E−01	5.65E+00	**5.05E−03**^{CFUR}
$CEC05_8$	2.13E+01	2.13E+01	2.13E+01	2.14E+01	**2.08E+01**^{CFUR}
$CEC05_9$	5.57E+02^{L}	4.71E+02^{L}	7.09E+02^{L}	**2.00E+02**^{L}	7.62E+02
$CEC05_{10}$	9.20E+02^{L}	1.12E+03^{L}	1.04E+03^{L}	**6.95E+02**^{L}	1.32E+03
$CEC05_{11}$	**1.34E+02**	1.40E+02	**1.31E+02**^{L}	1.34E+02	1.36E+02
$CEC05_{12}$	**5.01E+05**	5.87E+05	1.86E+06	6.54E+05	7.07E+05^{R}
$CEC05_{13}$	**6.12E+01**^{L}	6.81E+01^{L}	9.61E+01^{L}	8.32E+01^{L}	1.35E+02
$CEC05_{14}$	**4.65E+01**	4.71E+01	4.70E+01	4.69E+01	4.66E+01^{FUR}

The best solution found by the methods has been shown in bold

Table 7 Comparison of the methods when $d = 500$ and 1000 (CEC08 was used as the benchmark)

Function	$d = 500$			$d = 1000$		
	COPSO	UGPSO	LcRiPSO	COPSO	UGPSO	LcRiPSO
$CEC08_1$	2.94E+06	1.31E+06	**7.19E−07**^{CU}	6.00E+06	5.93E+06	**1.30E+01**^{CU}
$CEC08_2$	1.77E+02	1.62E+02	**8.41E+01**^{CU}	1.84E+02	1.86E+02	**9.41E+01**^{CU}
$CEC08_3$	3.22E+12	3.48E+11	**1.17E+03**^{CU}	6.94E+12	7.04E+12	**6.28E+04**^{CU}
$CEC08_4$	1.20E+04	1.29E+04	**4.51E+03**^{CU}	2.46E+04	2.39E+04	**8.92E+03**^{CU}
$CEC08_5$	2.46E+04	2.19E+04	**1.10E−03**^{CU}	5.34E+04	5.25E+04	**5.99E−01**^{CU}
$CEC08_6$	2.14E+01	2.14E+01	**1.93E+01**^{CU}	2.15E+01	2.15E+01	**1.94E+01**^{CU}

The best solution found by the methods has been shown in bold

Finally, LcRiPSO is compared with UGPSO and COPSO when they are applied to CEC08 benchmark functions with 500 and 1000 dimensions.

Table 7 indicates that for both 500 and 1000 dimensions, in all cases the performance of LcRiPSO is significantly better than the others. In fact, these methods could not improve the initial solutions at all, while the proposed method could still suggest improvements in the initial solutions.

5.7 Test 4, Rotation

In order to test the ability of the proposed method in finding solutions when the search space is rotated, we applied LcRiPSO, COPSO, and RPSO to $CEC05_{2,4,5,6,9}$ functions (selected randomly) when they are rotated randomly (the rotation matrix was the same for all methods

Table 8 Effect of rotation on the performance of the algorithms

Function	COPSO		RPSO		LcRiPSO	
	Original	Rotated	Original	Rotated	Original	Rotated
$CEC05_2$	9.66E−27	6.37E−27	1.21E−27	1.21E−26	1.03E−28	1.37E−28
$CEC05_4$	**1.49E+03***	5.49E+03	4.41E+01	1.42E+03	2.61E−28	3.82E−28
$CEC05_5$	4.04E+03	**2.34E+03***	1.41E+03	9.23E+02	1.50E+03	1.74E+03
$CEC05_6$	**2.27E+01***	3.39E+03	3.49E+03	3.34E+03	2.60E+03	3.99E+03
$CEC05_9$	**1.51E+01***	3.18E+01	8.14E+01	7.17E+01	2.69E+01	2.68E+01

In the cases where the performance of the algorithms in optimizing the rotated function is significantly different from optimizing the original function, the better solution has been indicated by a star
The best solution found by the methods has been shown in bold

and all runs). Also, because the rotation of the coordinate could potentially move the optimal solutions out of the boundaries, the boundaries were extended in our experiments by 5 times. The number of dimensions was set to 10, the swarm size was set to 10, and the number of FE was set to 5000.

In Table 8, it is clear that the performance of COPSO is significantly affected (indicated by a star, *) in 4 functions out of 5 when the search space is rotated. However, the performance of LcRiPSO and RPSO is not significantly affected when the search space is rotated. This suggests that COPSO is sensitive to rotation of the search space while LcRiPSO and RPSO are not.

6 Conclusion and future work

Six well-known issues in the standard particle swarm optimizer were analyzed in this paper. These issues were: stagnation (when all particles collapse on the same location, the algorithm does not improve anymore), dimensional stagnation (when particles and personal best vectors share one dimension, particles oscillate in that dimension), swarm size (performance of the algorithm radically impaired when the swarm size is small), local convergence (the algorithm cannot converge to better solutions), problem scale (when the scale of the problem grows, the algorithm cannot improve the initial solutions and stops in the initial solution), and rotation invariance (the algorithm is sensitive to rotation of the space). Some existing PSO variants that address these issues were summarized from the experimental and theoretical points of view. It was observed that none of the listed PSO variants can address all of these issues at the same time. A new general form of the velocity update rule in PSO was proposed (Eq. 12b) in which a new component was added. This new component is a function that is applied to the personal best of each particle in each iteration. The characteristics of the introduced function were analyzed in order to guarantee to address all six mentioned issues. It was proven that the proposed method can address all of these issues if the introduced function satisfies two conditions. To show that such a function exists, one specific designation was proposed (Eq. 18). The proposed method (with the specific function) was applied to some standard benchmark functions to test its abilities in addressing six mentioned issues. These experiments showed that the proposed method is the only one (amongst other tested PSO variants) that can address all six issues at the same time. Also, the parameters of the method were set by some experiments. In order to test the overall performance of the proposed method, it

was applied to 20 standard benchmark functions (CEC05 and CEC08) with different number of dimensions (from 2 to 1000 dimensions). Results showed that the proposed method has better overall performance in comparison to other tested PSO variants. However, there are some open questions that can be considered for future studies. The general idea behind the proposed method was to address some issues in the algorithmic level. It would be valuable to compare other methods that address the same issues in meta-algorithmic level, e.g., restarting the particles to address stagnation, with the proposed method (Garcia-Nieto and Alba 2011). The proposed specific model for the function f can be revised and other, possibly better, functions can be considered in the algorithm instead. For example, one can design a function with adaptive parameters to adapt the behavior of the algorithm according to some features (e.g., ruggedness, neutrality) of the landscape. Also, the global convergence properties of the algorithm can be investigated. For example, the analysis conducted in Van den Bergh and Engelbrecht (2010) for the global convergence properties of a PSO variant can be also performed for the method proposed in this paper. Moreover, parameters presented in the proposed model originated from other types of PSO. Effects of changing these parameters on the performance of the proposed method can be studied in details in future. The additional parameter introduced to the specific model was set manually and it would be worthwhile to experiment with an adaptive approach. Effects of changing the topology of the swarm on the performance of the proposed method is another topic for further studies (e.g., see Montes de Oca and Stutzle 2008 for an analysis of FIPS). Finally, consideration of the runtime (Witt 2009) and first hitting time (Lehre and Witt 2013) constitute another possibility of analysis of the proposed approach.

Acknowledgments The authors would like to extend their great appreciation to Maris Ozols, Luigi Barone, Frank Neumann, and Markus Wagner for constructive comments and discussions that have helped us to improve the quality of the paper. This work was partially funded by the ARC Discovery Grant DP130104395 and by Grant N N519 5788038 from the Polish Ministry of Science and Higher Education (MNiSW).

Appendix 1

A PSO method is locally convergent if

$$\forall i \lim_{t\to\infty} P\left(\vec{p}_t^{\,i} \in R_\varepsilon\right) = 1, \tag{19}$$

i.e., the probability that the personal best of each particle i, $\vec{p}_t^{\,i}$, is in the optimality region R_ε approaches 1 when the iteration number t approaches infinity.

In this appendix we prove the following

Theorem *If the function f in the velocity update rule of the LcRiPSO (Eq. 12b) satisfies the condition:*

$$\forall \vec{y} \in S \exists A_y \subseteq S \, \forall \vec{z} \in A_y \quad \forall \delta > 0 \quad P\left(|f(\vec{y}) - \vec{z}| < \delta\right) > 0, \tag{20}$$

then the LcRiPSO algorithm is locally convergent. In the condition 20, $\vec{y}$ is an arbitrary point in the search space S, A_y is an open set which contains $\vec{y}$, $\vec{z}$ is an arbitrary point in A_y, δ is a positive value.

Equation 20 can be explained as follows: for all $\vec{y}$ in the search space, there exist an open set A_y in the search space that contains $\vec{y}$ such that for every point $\vec{z}$ in this open set, for every real value $\delta > 0$, the point $f(\vec{y})$ is closer than δ to $\vec{z}$ with non-zero probability. Informally,

20 states that the value of $f(\vec{y})$ can map the point $\vec{y}$ to any point in the open set A_y with non-zero probability.

If the function f satisfies condition 20, the personal best of at least one of the particles converges to the optimality region of the objective function with probability one.

Before we proceed, let us define a general form for a stochastic algorithm (GSA) (Solis and Wets 1981):

Algorithm GSA:

1) **initialize p_0 from the search space S and set $t = 1$**
2) **generate a random sample x_t from S**
3) **generate the candidate solution $p_t = D(p_{t-1}, x_t)$, set $t = t + 1$, and go to 2**

where $D(a, b)$ is an operator which selects one of a or b. There are three important components in GSA: (1) a random sample x_t, (2) a candidate solution p_t, and (3) an update procedure of p_t (the operator D). We investigate the local convergence condition (condition 19) for GSA. We introduce two conditions C1 and C2, and we prove that if GSA satisfies C1 and C2, then GSA is locally convergent. We then show that each particle i in the LcRiPSO is a specific model of GSA. Hence, if C1 and C2 are satisfied for each particle in the LcRiPSO then the algorithm is locally convergent. Finally, we prove that all particles in the LcRiPSO satisfy C1 and C2. This would complete the proof of local convergence for the LcRiPSO.

Let us start with defining conditions C1 and C2.

Condition C1: GSA satisfies condition C1 if:

$$p_t = D(p_{t-1}, x_t) = \begin{cases} x_t & \text{if } F(x_t) < F(p_{t-1}) - \varepsilon_0 \\ p_{t-1} & \text{otherwise,} \end{cases}$$

where ε_0 is a positive value that is smaller than or equal to ε (ε in the definition of $R_{\varepsilon i}$). This means that the new solution x_t should be better than p_{t-1} at least by the constant ε_0 to update p_t. In computer simulations, we can set ε_0 to the smallest possible float/double value (Matyas 1965).

Condition C2: GSA satisfies the condition C2 if:

$$\exists \varepsilon > 0 \exists \eta > 0 \exists \delta \in (0, 1] \forall t \geq 0 \exists t' > 0 P(F(p_{t+t'}) \leq F(p_t) - \eta) > \delta \quad \text{or} \quad p_t \epsilon R_\varepsilon,$$

i.e., $p_{t+t'}$ is better than p_t at least by η in terms of the objective value $F(.)$. In other words, the probability that the $p_{t+t'}$ is better than p_t(in terms of objective value F) at least by the value η is larger than δ unless p_t is already in the optimality region. We will prove (lemma 1) that if both C1 and C2 are satisfied, GSA is locally convergent.

Lemma 1 *If GSA satisfies conditions C1 and C2, GSA is locally convergent.*

Proof Let us define

$$A(t, t') = \begin{cases} \text{true} & F(p_{t+t'}) \leq F(p_t) - \eta \\ \text{false} & \text{otherwise.} \end{cases}$$

Then, the negation of $A(t, t')$, not $A(t, t')$, is:

$$\bar{A}(t, t') = \begin{cases} \text{true} & F(p_{t+t'}) > F(p_t) - \eta \\ \text{false} & \text{otherwise.} \end{cases} \tag{21}$$

So, according to the C2, we can say that for any t, there exist t' such that the probability of $p_{t+t'}$ being not better than p_t is smaller than (or equal to) $1 - \delta$ (the complement of the probability in the condition C2):

$$P\left(\bar{A}(t, t')\right) \leq 1 - \delta.$$

Let us consider now a sequence of k successive occurrences of $\bar{A}\left(t, t'\right)$: $\bar{A}\left(t + t'_0, t'_1\right)$, $\bar{A}\left(t + t'_0 + t'_1, t'_2\right), \ldots, \bar{A}\left(t + t'_0 + t'_1 \ldots + t'_{k-1}, t'_k\right)$, i.e., $p_{t+t'_0+t'_1\ldots+t'_{k-1}+t'_k}$ being not better than p_t. Note that (based on C1) p_{t+1} is not worse than p_t, so, if $p_{t+t'_0+t'_1\ldots+t'_{k-1}+t'_k}$ is not better than p_t, then none of p_l is better than p_t for all $l \in \left[t, t + t'_0 + t'_1 \ldots + t'_{k-1} + t'_k\right]$. Hence, for any t and for any number of steps k, the probability of $p_{t+t'_0+t'_1\ldots+t'_{k-1}+t'_k}$ being not better than p_t is calculated by:

$$P\left(\bar{A}\left(t + t'_0, t'_1\right)\right) * P\left(\bar{A}\left(t + t'_0 + t'_1, t'_2\right)\right) * \ldots * P\left(\bar{A}\left(t + t'_0 + t'_1 \ldots + t'_{k-1}, t'_k\right)\right)$$
$$= \prod_{l=1}^{k} P\left(\bar{A}\left(t + \sum_{j=0}^{l-1} t'_j, t'_l\right)\right) \leq (1-\delta)^k,$$

where t'_0 is 0. Therefore, the probability that at least one of $\{p_{t+1}, p_{t+2}, \ldots, p_{t+t'_0+t'_1\ldots+t'_{k-1}+t'_k}\}$ is better than p_t is given by:

$$1 - \prod_{l=1}^{k} P\left(\bar{A}\left(t + \sum_{j=0}^{l-1} t'_j, t'_l\right)\right) > 1 - (1-\delta)^k.$$

As k grows, the right hand side of the inequality approaches 1. Thus, the probability that at least one of $\{p_{t+1}, p_{t+2}, \ldots, p_{t+t'_0+t'_1\ldots+t'_{k-1}+t'_k}\}$ is better than p_t by at least η, grows to 1. Let us denote the event of p_t becoming better by at least η in the next $t'_0 + t'_1 \ldots + t'_{k-1} + t'_k$ iterations by $A^*(t, t'_k)$. The probability of this event approaches 1 as the number of steps k grows.

Now, with further iterations of the GSA algorithm (i.e., as t grows), a sequence of occurrences of $A^*(t, t'_k)$ takes place—every occurrence of $A^*(t, t'_k)$ results in improvement by at least η of the personal best vector with probability 1. As the number of such occurrences $A^*(t, t'_k)$ grows, p_t will arrive at the optimality region with probability 1. This completes the proof for the lemma 1. □

Clearly, each particle i in the LcRiPSO is a specific model of the GSA. In fact, the personal best of the particle i ($\vec{p}_t^{\,i}$) corresponds to a candidate solution p_t in GSA. Further, the personal best of the particle i is updated by:

$$\vec{p}_t^{\,i} = \begin{cases} \vec{x}_t^{\,i} & \text{if } F\left(\vec{x}_t^{\,i}\right) < F\left(\vec{p}_{t-1}^{\,i}\right) - \varepsilon_0 \\ \vec{p}_{t-1}^{\,i} & \text{otherwise,} \end{cases} \tag{22}$$

where ε_0 is the desired precision of the optimality region (see the condition C1) that can be set to the smallest possible value in computer simulations. Thus, the updating procedure 22 in LcRiPSO for the particle i corresponds to the operator D in GSA. Also, the current position of the particle i ($\vec{x}_t^{\,i}$) corresponds to the random sample x_t in GSA. As particle i contains all three components in GSA, i.e., the current position in a particle is corresponding to the random sample, and the personal best of a particle is corresponding to candidate solution, and, finally, Eq. 23 of a particle is corresponding to operator D, we conclude that each particle i in LcRiPSO is a specific model of GSA. Because each particle i in LcRiPSO is a specific model of GSA, if we prove that each particle i satisfies C1 and C2, we would prove also that LcRiPSO is locally convergent (recall that, based on Eq. 19, if all particles converge to the optimality region, the PSO method is locally convergent).

Before we start the proof of the theorem, let us analyze how the position of the particle i is updated. The position and velocity update rule in LcRiPSO for the ith particle at the time t is written as:

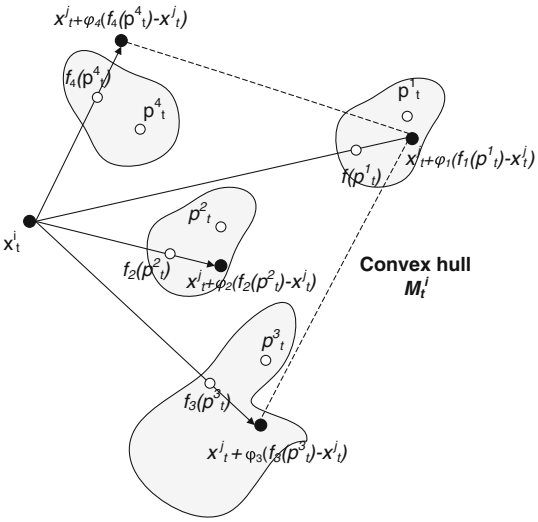

Fig. 8 A particular example for the convex hull M_t^i where the size T_t^i is 4. Five points that the convex hull has been defined by have been shown in the figure by *black circles*. The *gray* areas show the area $A_{\vec{p}t^k}$ for each $\vec{p}_t^k$. Note that $f(\vec{p}_t^k)$ can be anywhere in $A_{\vec{p}_t^k}$

$$\vec{x}_{t+1}^i = \vec{x}_t^i + \vec{V}_{t+1}^i, \tag{23}$$

$$\vec{V}_{t+1}^i = \omega \vec{V}_t^i + \vec{v}_t^i, \tag{24}$$

$$\vec{v}_t^i = \sum_{k \in T_t^i} \varphi_k r_{kt}^i \left(f_k(\vec{p}_t^k) - \vec{x}_t^i \right), \tag{25}$$

where T_t^i is the set of indexes of the particles which contribute to the velocity update rule of the particle i, $\vec{p}_t^k$ is the personal best of the kth particle, and φ_k and ω are constants. Note that based on the definition of T_t^i, we always assume that $i \in T_t^i$ (i.e., the particle i always contributes to its own velocity update rule). $\vec{p}_t^i$ is updated using Eq. 22. By combining Eqs. 23 and 24, we get

$$\vec{x}_{t+1}^i = \vec{x}_t^i + \omega \vec{V}_t^i + \vec{v}_t^i. \tag{26}$$

Also, according to Eq. 23,

$$\vec{V}_t^i = \vec{x}_t^i - \vec{x}_{t-1}^i. \tag{27}$$

By combining Eqs. 26 and 27 we get

$$\vec{x}_{t+1}^i = \vec{x}_t^i + \vec{v}_t^i + \omega(\vec{x}_t^i - \vec{x}_{t-1}^i). \tag{28}$$

Let us pay attention to the first two components of this formula, $\vec{x}_t^i$ and $\vec{v}_t^i$. Since calculation of $\vec{v}_t^i$ (see 25) involves random values r_{kt}^i, then $\vec{x}_t^i + \vec{v}_t^i$ is a random point. Now we introduce a new construct that plays an important role in our proof. For the particle i, we define a convex hull M_t^i that is defined by $card(T_t^i)+1$ vertices: $\vec{x}_t^i, \vec{x}_t^i + \varphi_k \left(f_k(\vec{p}_t^k) - \vec{x}_t^i \right)$ for all $k \in \left\{ T_t^i \right\}$. Figure 8 shows an example of this convex hull for the particle i.

We are now going to introduce another lemma (lemma 2) that is essential for proving the satisfaction of C2 by LcRiPSO.

Lemma 2 *For every convex hull M which is defined by the points {Y_1, Y_1+Y_2,..., Y_1+Y_n}, there exist* $\{r_2, \ldots, r_n\}$, $r_k \in [0, 1]$ *for* $k = 2, \ldots, n$, *such that any point m inside M can be represented by:* $m = Y_1 + \sum_{k=2}^{n} r_k Y_k$.

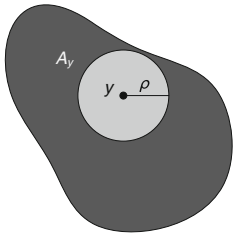

Fig. 9 $f(\vec{y})$ can be any point in the *light gray* area. The *dark* area shows the open set A_y

Proof Let us define $Z_1 = Y_1$ and $Z_k = Y_1 + Y_k$ for $k = 2, \ldots, n$. If we translate the origin of the coordinate system to Y_1, then the vertices for M' (the convex hull M in the new coordinate system) are $\{Z_1 - Y_1, Z_2 - Y_1, \ldots, Z_n - Y_1\}$ that is $\{Z'_1, Z'_2, \ldots, Z'_n\}$ where Z'_1 is in the origin of the new coordinate system. Any arbitrary point m' inside M' (note that $m' = m - Y_1$ where m is a point inside M) can be represented by a *convex combination* of all vertices in M' (Rockafellar 1996). In other words:

$$\exists \{r_1, r_2, \ldots, r_n\}, r_k \in [0, 1] \text{ for all } k = 1, 2, \ldots, n, m' = \sum_{k=1}^{n} r_k Z'_k.$$

Note that, according to the definition of convex combination, $\{r_1, r_2, \ldots, r_n\}$ have the property $\sum_{k=1}^{n} r_k = 1$. As Z'_1 is the origin, it is clear that $Z'_1 = r_1 Z'_1$ for anyr_1. Thus, we get

$$\exists \{r_2, \ldots, r_n\}, r_k \in [0, 1] \text{ for all } k = 2, \ldots, n, m' = Z'_1 + \sum_{k=2}^{n} r_k Z'_k.$$

By substituting $m' = m - Y_1$ and $Z'_k = Z_k - Y_1$, any point m inside M can be represented by $m - Y_1 = Z_1 - Y_1 + \sum_{k=2}^{n} r_k (Z_k - Y_1) = \sum_{k=2}^{n} r_k Y_k$. Hence, there exist $\{r_2, \ldots, r_n\}, r_k \in [0, 1] \, for k = 2, \ldots, n$ such that $m = Y_1 + \sum_{k=2}^{n} r_k Y_k$. □

According to the lemma 2, for any point $\vec{m}_t^i$ in the convex hull M_t^i, there exist r_{kt}^i (for all $k \in T_t^i$) that $\vec{m}_t^i = \vec{x}_t^i + \vec{v}_t^i$ (see Eq. 25). Thus, we can write 28 as

$$\vec{x}_{t+1}^i = \vec{m}_t^i + \omega(\vec{x}_t^i - \vec{x}_{t-1}^i), \tag{29}$$

where $\vec{m}_t^i$ is a random point in M_t^i. Therefore, the point $\vec{x}_{t+1}^i$ is a random point from the subspace $(M_t^i + \omega(\vec{x}_t^i - \vec{x}_{t-1}^i)$.

We introduce two simple observations that are used in the main proof.

Observation (a) For any point $\vec{y}$ in the search space S, there exists a hyper-sphere with the center $\vec{y}$ and the radius ρ (we use the notation $n_\rho(\vec{y})$ for the set of all points in a hyper-sphere with the center $\vec{y}$ and radius ρ) that $f(\vec{y})$) can be arbitrarily close to any point in that hyper-sphere with non-zero probability. In other words, $\forall \vec{y} \in S \; \exists \rho > 0 \; \forall \vec{z} \in n_\rho(\vec{y}) \; \forall \varepsilon > 0 \; P(|f(\vec{y}) - \vec{z}| < \varepsilon) > 0$ where $n_\rho(\vec{y})$ is the hyper-sphere with the radius ρ centered at $\vec{y}$ (Fig. 9).

This is because A_y (in 20) is an open set which contains $\vec{y}$, and as $n_\rho(\vec{y})$ is a sphere-neighborhood of $\vec{y}$, all points in $n_\rho(\vec{y})$ are in A_y. Thus, $n_\rho(\vec{y})$ is a subset of A_y. According to the condition 20, $f(\vec{y})$ can be arbitrarily close to any point in A_y with non-zero probability. Hence, $f(\vec{y})$ has non-zero probability to be arbitrarily close to any point in $n_\rho(\vec{y})$.

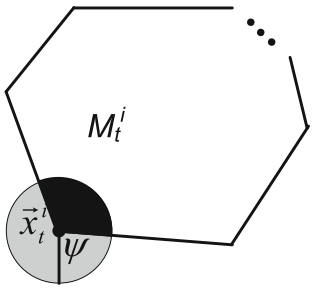

Fig. 10 The intersection between $n_\psi\left(\vec{x}_t^i\right)$ and M_t^i is non-empty. The *dark area* shows the intersection. Note that the shape of M_t^i depends on $f_k(\vec{p}_t^k)$ and $\vec{x}_t^i$, while it is always convex

Observation (b) For every particle i at every iteration t, for every $\psi > 0$, the convex hull M_t^i has non-empty intersection with $n_\psi\left(\vec{x}_t^i\right)$. In fact,

$$\forall i\ \forall t\ \forall \psi > 0\ M_t^i \cap n_\psi\left(\vec{x}_t^i\right) \neq \emptyset.$$

This is because $\vec{x}_t^i \in M_t^i$ (note that based on the definition of M_t^i, $\vec{x}_t^i$ is always a point in M_t^i), hence $M_t^i \cap n_\psi\left(\vec{x}_t^i\right) \neq \emptyset$. Also, note that this intersection reduces to only one point $\vec{x}_t^i$ when $M_t^i = \left\{\vec{x}_t^i\right\}$ where $\left\{\vec{x}_t^i\right\} = \left\{f_k\left(\vec{p}_t^k\right)\right\}$ forall$k \in \left\{T_t^i\right\}$. Also, this intersection is a sub-space of $n_\psi\left(\vec{x}_t^i\right)$ when $M_t^i - \left\{\vec{x}_t^i\right\} \neq \emptyset$. Figure 10 shows this intersection when $M_t^i - \left\{\vec{x}_t^i\right\} \neq \emptyset$.

This observation implies also that for all $\psi > 0$ there exist a vector $\vec{a}$ which its length is shorter than ψ and the vector $\vec{x}_t^i + \vec{\alpha}$ is a member of M_t^i (note that $n_\psi\left(\vec{x}_t^i\right)$ has non-empty intersection with M_t^i). Also, according to the lemma 2, $\vec{m}_t^i$ (a random point in M_t^i) can be any point in $M_t^i \cap n_\psi\left(\vec{x}_t^i\right)$ with non-zero probability for all ψ. Note that, in this case, $\vec{m}_t^i$ can be represented by $\vec{x}_t^i + \vec{\alpha}$ where $\vec{\alpha}$ is a random vector in $M_t^i \cap n_\psi\left(\vec{x}_t^i\right)$ and its length is less than ψ.

Lemma 3 *In the LcRiPSO algorithm, if the function f satisfies the condition 20, then*

$$\forall \omega \in (0,\ 1)\ \exists \beta > 0 \forall i \forall t \exists h > \frac{\beta}{1-\omega} \forall \vec{z} \in n_h\left(\vec{p}_t^i\right)\ P\left(\vec{z} \in M_t^i\right) > 0.$$

In other words, for every value ofω in the interval [0, 1], there exists $\beta > 0$ that for every particle i at every iteration t, every point in $n_h\left(\vec{p}_t^i\right)$ has non-zero probability to be in the convex hull M_t^i.

Proof According to the Observation (a), for all $\vec{y}$ in the search space, there exists a ρ where $f\left(\vec{y}\right)$ can be arbitrarily close to any point in $n_\rho\left(\vec{y}\right)$ with non-zero probability. Based on the definition of M_t^i, $f(\vec{p}_t^i)$ is in M_t^i. This implies that any point in $n_\rho\left(\vec{p}_t^i\right)$ has non-zero probability to be in M_t^i. Also, for any $\rho > 0$, for all ω, there exist a β in a way that $\frac{\beta}{1-\omega} = \rho$. Thus, by considering $h = \rho$, any point in $n_h\left(\vec{p}_t^i\right)$ has non-zero probability to be in M_t^i. This completes the proof for the lemma 3. □

Now, we are ready to prove the main theorem.

Theorem *If the function f in the velocity update rule of LcRiPSO (Eq. 12b) satisfies condition:*

$$\forall \vec{y} \in S \exists A_y \subseteq S \forall \vec{z} \in A_y \forall \delta > 0 P\left(\left|f\left(\vec{y}\right) - \vec{z}\right| < \delta\right) > 0, \tag{30}$$

then the algorithm LcRiPSO (the proposed PSO algorithm which uses Eq. 12b for velocity update rule and the standard position update rule) is locally convergent.

Proof We will show that any particle i in LcRiPSO satisfies conditions C1 and C2. As the personal best of each particle is updated by Eq. 23, the condition C1 is satisfied for each particle. Now, for any iteration t, let us consider $t' > 0$ additional iterations. There are two cases to consider:

$$\text{case (1) } F\left(\vec{p}^{\,i}_{t+t'}\right) < F\left(\vec{p}^{\,i}_{t}\right)$$
$$\text{case (2) } F\left(\vec{p}^{\,i}_{t+t'}\right) = F\left(\vec{p}^{\,i}_{t}\right)$$

Case (1) implies that there exist $0 < \tau \leq t'$ that $F\left(\vec{p}^{\,i}_{t+\tau}\right) < F\left(\vec{p}^{\,i}_{t}\right)$. According to 23, if $F\left(\vec{p}^{\,i}_{t+\tau}\right)$ is better than $F\left(\vec{p}^{\,i}_{t}\right)$, it is better by at least ε_0. By setting $\varepsilon_0 = \eta$ in the condition C2, this condition is satisfied for $\tau = t'$. This completes the proof for this case. Let us continue with the case (2).

Case (2) implies that $\vec{p}^{\,i}_{t} = \vec{p}^{\,i}_{t+\tau}$ for all $0 < \tau \leq t'$. We will show that in this case, for all $0 < \omega < 1$ for all $\psi > 0$ there exist h>0 for each particle i at all iteration t, there exist t'>0 that $\vec{x}^{\,i}_{t+t'}$ can be any point[10] in $n_h\left(\vec{p}^{\,i}_{t}\right)$ with non-zero probability. Hence, if there exists a point in $n_h\left(\vec{p}^{\,i}_{t}\right)$ that is better than $\vec{p}^{\,i}_{t}$ by at least η, there is a non-zero probability that $\vec{x}^{\,i}_{t+t'}$ is that point, and hence, $\vec{p}^{\,i}_{t+t'}$ is updated to $\vec{x}^{\,i}_{t+t'}$. As $\vec{p}^{\,i}_{t+t'}$ is better than $\vec{p}^{\,i}_{t}$ by at least η with non-zero probability, C2 is satisfied. We also show that if such a point does not exist then $\vec{p}^{\,i}_{t}$ is already in the optimality region. This would complete the proof for the case (2).

According to the Observation (b), for all ψ, $\vec{m}^{\,i}_{t} = \vec{x}^{\,i}_{t} + \vec{\alpha}_0$ with non-zero probability, where $|\vec{\alpha}_0| < \psi$. From the updating equation for $\vec{x}^{\,i}_{t+1}$ (Eq. 29), we get:

$$\vec{x}^{\,i}_{t+1} = \vec{x}^{\,i}_{t} + \vec{\alpha}_0 + \omega\left(\vec{x}^{\,i}_{t} - \vec{x}^{\,i}_{t-1}\right), \tag{31}$$

with non-zero probability. Similarly, for $\vec{x}^{\,i}_{t+2}$:

$$\begin{aligned}\vec{x}^{\,i}_{t+2} &= \vec{m}^{\,i}_{t+1} + \omega\left(\vec{x}^{\,i}_{t+1} - \vec{x}^{\,i}_{t}\right) = \vec{x}^{\,i}_{t+1} + \vec{\alpha}_1 + \omega\left(\vec{x}^{\,i}_{t} + \omega\left(\vec{x}^{\,i}_{t} - \vec{x}^{\,i}_{t-1}\right) + \vec{\alpha}_0 - \vec{x}^{\,i}_{t}\right)\\ &= \vec{x}^{\,i}_{t+1} + \vec{\alpha}_1 + \omega\left(\omega\left(\vec{x}^{\,i}_{t} - \vec{x}^{\,i}_{t-1}\right) + \vec{\alpha}_0\right) = \vec{x}^{\,i}_{t+1} + \vec{\alpha}_1 + \omega\vec{\alpha}_0 + \omega^2\left(\vec{x}^{\,i}_{t} - \vec{x}^{\,i}_{t-1}\right),\end{aligned} \tag{32}$$

with non-zero probability. In general,

$$\vec{x}^{\,i}_{t+t'} = \vec{m}^{\,i}_{t+t'-1} + \omega^{t'}\left(\vec{x}^{\,i}_{t} - \vec{x}^{\,i}_{t-1}\right) + \sum_{l=0}^{t'-1}\omega^{t'-1-l}\vec{\alpha}_l, \tag{33}$$

with non-zero probability, where $|\vec{\alpha}_l| < \psi$ for all l. For simplicity, we re-write the last equation as:

$$\vec{x}^{\,i}_{t+t'} = \vec{m}^{\,i}_{t+t'-1} + \vec{\lambda}_{t',\omega} + \vec{\zeta}_{t',\psi}, \tag{34}$$

where $\vec{\zeta}_{t',\psi} = \sum_{l=0}^{t'-1}\omega^{l}\vec{\alpha}_l$, $|\vec{\alpha}_l| < \psi$ and $\vec{\lambda}_{t',\omega} = \omega^{t'}\left(\vec{x}^{\,i}_{t} - \vec{x}^{\,i}_{t-1}\right)$.[11] At the iteration $t + t'$, the random sample $\vec{m}^{\,i}_{t+t'-1}$ is transformed by the vector $\vec{\lambda}_{t',\omega} + \vec{\zeta}_{t',\psi}$.

[10] Note that, as all of the discussions in this paper are around continuous space, hitting a point is impossible. Hence, whenever we are using "a point" we refer to "arbitrarily close to a point."

[11] Note that both $\vec{\lambda}_{t',\omega}$ and $\vec{\zeta}_{t',\psi}$ are dependent on t and i as well, however, for simplicity, these two indexes are not mentioned in the usage of $\vec{\lambda}_{t',\omega}$ and $\vec{\zeta}_{t',\psi}$.

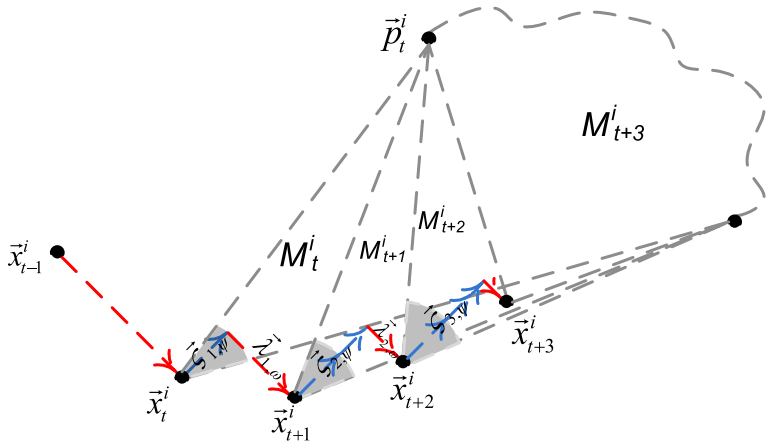

Fig. 11 The *blue* vectors show $\vec{\zeta}_{t',\psi}$. The *gray* areas are $M_{ti} \cap n_\psi\left(\vec{x}_t^{\,i}\right)$. The length of the *blue* vectors is always smaller than ψ. Also, the *red* vectors show the term $\vec{\lambda}_{t',\omega}$. It is clear that the *red* vectors become smaller in each iteration.

Let us analyze all three components of Eq. 34 (i.e., $\vec{m}_{t+t'-1}^{\,i}$, $\vec{\lambda}_{t',\omega}$, and $\vec{\zeta}_{t',\psi}$).

Analysis of $\vec{\lambda}_{t',\omega}$: Clearly, $\omega^{t'}$ becomes smaller as t' grows (recall that $0 \leq \omega < 1$ is essential for PSO to be locally convergent), thus $\left|\vec{\lambda}_{t',\omega}\right|$ becomes closer to zero as t' grows (note that the length of $\vec{x}_t^{\,i} - \vec{x}_{t-1}^{\,i}$ is constant).

Analysis of $\vec{\zeta}_{t',\psi}$: The longest possible vector that can be generated by this term is achieved if for all l, $\vec{\alpha}_l$ are in the same direction and their lengths are ψ (according to the Observation (b) the lengths of $\vec{\alpha}_l$ is at most ψ). Thus,

$$\left|\vec{\zeta}_{t',\psi}\right| = \left|\sum_{l=0}^{t'-1} \omega^l \vec{\alpha}_l\right| < \sum_{l=0}^{t'-1} \omega^l \psi = \psi \sum_{l=0}^{t'-1} \omega^{t'-1-l},$$

that is always smaller than $\psi \frac{\omega}{1-\omega}$. Hence,

$$\forall \omega \in (0,1)\, \forall \psi > 0 \forall i \forall t \forall t' > 0 \left|\vec{\zeta}_{t',\psi}\right| < \psi \frac{\omega}{1-\omega}.$$

Figure 11 shows the updating steps for $\vec{x}_t^{\,i}$ in three iterations (it shows the changes of $\vec{\zeta}_{t',\psi}$ and $\vec{\lambda}_{t',\omega}$. It is clear that the length of $\vec{\lambda}_{t',\omega}$ shrinks (red vectors). Also, the length of $\vec{\zeta}_{t',\psi}$ (blue vectors) in each iteration is the sum of all previous $\vec{\alpha}_l$ multiplied by $\omega^{t'-1-l}$ $(0 < l < t' - 1)$.

Based on the **Analysis of** $\vec{\zeta}_{t',\psi}$, it is clear that $\left|\vec{\zeta}_{t',\psi} + \vec{\lambda}_{t',\omega}\right| \leq \left|\vec{\zeta}_{t',\psi}\right| + \left|\vec{\lambda}_{t',\omega}\right| < \psi \frac{\omega}{1-\omega} + \left|\vec{\lambda}_{t',\omega}\right|$. Because $\vec{\lambda}_{t',\omega}$ shrinks as t′ grows (**Analysis of** $\vec{\lambda}_{t',\omega}$), we can say that

$$\forall \omega \in (0,1)\, \forall \psi > 0 \forall i \forall t \exists t' > 0 \left|\vec{\zeta}_{t',\psi} + \vec{\lambda}_{t',\omega}\right| < \psi \frac{\omega}{1-\omega}.$$

Also, as this statement is true *for all* ψ, the statement is still true if the value of ψ is smaller than the value of β in the lemma 3, i.e., there exists t' that $\left|\vec{\zeta}_{t',\beta} + \vec{\lambda}_{t',\omega}\right| < \beta \frac{\omega}{1-\omega}$, and, based on Observation (b), that $\vec{m}_t^{\,i}$ can be any point in the intersection $M_t^i \cap_\psi^n \left(\mathbf{x}_t^i\right) \neq \emptyset$ **for all** ψ. According to the lemma 3, there exists $h > \frac{\beta}{1-\omega} = \beta + \beta \frac{\omega}{1-\omega}$, obviously

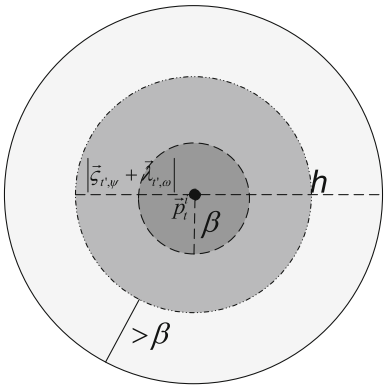

Fig. 12 The relation between $\left|\vec{\zeta}_{t',\psi}+\vec{\lambda}_{t',\omega}\right|$, h, and β. The *black dot* in the middle is $\vec{p}_{\mathrm{ti}}$, the largest circle is $\mathrm{n_h}\left(\vec{p}_{\mathrm{t}}^{\,\mathrm{i}}\right)$, the *dash-dotted circle* is a circle with the radius $\left|\vec{\zeta}_{t',\psi}+\vec{\lambda}_{t',\omega}\right|$, the *dashed circle* is a circle with the radius β. Note that $\mathrm{h} > \left|\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}\right|+\beta$

$h > \beta+\beta\frac{\omega}{1-\omega} > \beta\frac{\omega}{1-\omega} > \left|\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}\right|$. Hence, for all $h > \beta+\beta\frac{\omega}{1-\omega}$, it is also true that $h > \left|\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}\right|+\beta$. See Fig. 12 for the relation between $\left|\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}\right|$, β, and h.

Analysis of $\vec{m}^i_{t+t'-1}$: Based on the lemma 3, for every value ofω in the interval [0, 1], there exist $\beta > 0$ for every particle i at every iteration t there exist $h > \frac{\beta}{1-\omega}$ such that $n_h\left(\vec{p}_t^{\,i}\right)$ is in the convex hull M_t^i. As $\vec{m}^i_{t+t'-1}$ can be any point in M_t^i with non-zero probability (lemma 2), it can be any point in $n_h\left(\vec{p}_t^{\,i}\right)$ as well with non-zero probability.

Based on Eq. 34, $\vec{m}^i_{t+t'-1}$ is transformed by the vector $\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}$ to generate $\vec{x}^i_{t+t'}$. As $\vec{m}^i_{t+t'-1}$ is selected from $n_h\left(\vec{p}_t^{\,i}\right)$ with non-zero probability (**Analysis of $\vec{m}^i_{t+t'-1}$**, and because $h > \left|\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}\right|+\beta$, the point $\vec{x}^i_{t+t'} = \vec{m}^i_{t+t'-1}+\vec{\zeta}_{t',\beta}+\vec{\lambda}_{t',\omega}$ can be any point in $n_h\left(\vec{p}_t^{\,i}\right)$ with non-zero probability as well.

There are two possible cases for the points in $n_h\left(\vec{p}_t^{\,i}\right)$:

- there exists a point $\vec{p}$ in $n_h\left(\vec{p}_t^{\,i}\right)$ that is better than $\vec{p}_{\mathrm{ti}}$ by at least ε_0
- there is no point in $\mathrm{n_h}\left(\vec{p}_{t^{\mathrm{i}}}\right)$ that is better than $\vec{p}_{t^i}$ by at least ε_0

In the first case, $\vec{x}^i_{t+t'}$ has non-zero probability to be $\vec{p}$, which satisfies C2 as $F(\vec{p}) < F(\vec{p}_t^{\,i})-\varepsilon_0$. In the second case, $\vec{p}_t^{\,i}$ is already in the optimality region as the objective value of $\vec{p}_t^{\,i}$ is better than all points in $n_h\left(\vec{p}_t^{\,i}\right)$ by ε_0. Thus, the particle i satisfies C2, which implies that the particle i converges to the optimality region. As this particle was an arbitrary particle, all particles converge to the optimality region. This implies that the algorithm is locally convergent. □

Appendix 2

Theorem *it is proven that LcRiPSO is rotation, scale, and translation invariant if*

$$\forall s \in R \forall Orth\,(Q)\,\forall \vec{b}, \vec{y} \in R^d\, sQf(\vec{y})+\vec{b} = \hat{f}(sQ\vec{y}+\vec{b}).$$

Proof It is well-known that an algorithm is rotation, scalar, and translation invariant (RST invariant) if for all $t > 0$ $\hat{x}^i_{t+1} = sQ\vec{x}^i_{t+1}+\vec{b}$ for any scalar s, orthogonal matrix Q, and vector $\vec{b}$ (This is called *general RST invariant condition*), and $\hat{x}^i_{t+1}$is the position vector in the rotated, scaled, and translated space (Wilke et al. 2007b; Spiegel 1959) at iteration $t+1$.

Let us re-write the position and velocity update rules of the proposed method as follows:

$$\vec{x}^i_{t+1} = \vec{x}^i_t + \vec{V}^i_{t+1}, \tag{35}$$

$$\vec{V}^i_{t+1} = \omega \vec{V}^i_t + \vec{v}^i_t, \tag{36}$$

$$\vec{v}^i_t = \sum_{k \in T^i_t} \varphi_k r^i_{kt} \left(f_k(\vec{p}^k_t) - \vec{x}^i_t \right). \tag{37}$$

Note that, this notation is algebraically the same as the original notation. By calculating the left hand side ($\hat{x}^i_{t+1}$) and right hand side ($sQ\vec{x}^i_{t+1} + \vec{b}$) of the general RST invariant condition for the position update rule of the proposed method (Eq. 35), Eqs. 22 and 23 respectively emerge:

$$\hat{x}^i_{t+1} = \hat{x}^i_t + \hat{V}^i_{t+1} = sQ\vec{x}^i_t + \vec{b} + \hat{V}^i_{t+1}, \tag{38}$$

$$sQ\vec{x}^i_{t+1} + \vec{b} = sQ\left(\vec{x}^i_t + \vec{V}^i_{t+1}\right) + \vec{b} = sQ\vec{x}^i_t + \vec{b} + sQ\vec{V}^i_{t+1}. \tag{39}$$

By comparing Eqs. 38 and 39, it is obvious that the general RST invariant condition is satisfied if $\hat{V}^i_{t+1} = sQ\vec{V}^i_{t+1}$. The left hand side and right hand side of this equation for the proposed velocity update rule (Eq. 36) are calculated as Eqs. 40 and 41, respectively.

$$\hat{V}^i_{t+1} = \omega \hat{V}^i_t + \hat{v}^i_t = sQ\omega\vec{V}^i_t + \hat{v}^i_t, \tag{40}$$

$$sQ\vec{V}^i_{t+1} = sQ\omega\vec{V}^i_t + sQ\vec{v}^i_t. \tag{41}$$

Note that, if $\hat{V}^i_{t+1} = sQ\vec{V}^i_{t+1}$ then $\hat{V}^i_t = sQ\vec{V}^i_t$. By comparing Eqs. 40 and 41, it is obvious that $\hat{V}^i_{t+1} = sQ\vec{V}^i_{t+1}$ is satisfied iff $\hat{v}^i_t = sQ\vec{v}^i_t$. Thus, the general RST invariant condition is satisfied in the proposed method iff $\hat{v}^i_t = sQ\vec{v}^i_t$ for all Q and s, where $\hat{v}^i_t$ is the stochastic velocity that has been transformed (rotated, scaled, and translated). This condition is called *RST invariant condition*. The left hand side and right hand side of the RST invariant condition for Eq. 37 is written as

$$\hat{v}^i_t = \sum_{k \in T^i_t} \varphi_k r^i_{kt} \left(\hat{f}_k(\hat{p}^k_t) - \hat{x}^i_t \right) = \sum_{k \in T^i_t} \varphi_k r^i_{kt} \left(\hat{f}_k(Q\vec{p}^i_t + \vec{b}) - sQ\vec{x}^i_t - \vec{b} \right), \tag{42}$$

$$sQ\vec{v}^i_t = sQ\left(\sum_{k \in T^i_t} \varphi_k r^i_{kt} \left(f_k(\vec{p}^k_t) - \vec{x}^i_t \right) \right) = \sum_{k \in T^i_t} \varphi_k r^i_{kt} \left(sQf_k(\vec{p}^k_t) + \vec{b} - sQ\vec{x}^i_t - \vec{b} \right). \tag{43}$$

Equation 43 has been written according to this principle that, because r is a random scalar, we can say $Qr = rQ$ where Q is an orthogonal matrix. By comparing Eqs. 42 and 43, the condition $\hat{v}^i_t = sQ\vec{v}^i_t$ is satisfied if $sQf(\vec{y}) + \vec{b} = \hat{f}(sQ\vec{y} + \vec{b})$. □

Appendix 3

According to Gut (2009), if $N(\mu, V)$ (V is the covariance matrix and μ is the mean vector of the distribution) is rotated by Q, translated by b, and scaled by s, the mean of the new distribution is $b + sQ\mu$ and its covariance matrix is s^2QVQ^T, i.e., $\hat{N}(\hat{\mu}, V) = N(sQ\mu + b, s^2QVQ^T)$. Also, $N(\mu, V) = \mu + N(0, V)$

To study the condition $\hat{f}(\hat{x}) = sQf(x) + b$ for $f(\mu) = N(\mu, V)$, for the left hand side of the condition we have:

$$\hat{f}(\hat{x}) = \hat{N}(\hat{\mu}, V) = N(sQ\mu + b, s^2 QVQ^T) = sQ\mu + b + N(0, s^2 QVQ^T I). \quad (44)$$

For the right hand side of the condition:

$$\begin{aligned} sQf(x) + b &= sQN(\mu, V) + b = sQ\mu + b + sQN(0, V) \\ &= sQ\mu + b + N(0, s^2 QQ^T VI). \end{aligned} \quad (45)$$

By comparing Eqs. 45 and 44, in order to have the condition satisfied, we need to have $Q^T V = VQ^T$. This equation is satisfied if $V = aI$ where a is scalar and I is the identity matrix.

References

Bonyadi, M. R., Li, X., & Michalewicz, Z. (2013). A hybrid particle swarm with velocity mutation for constraint optimization problems. In *Genetic and evolutionary computation conference* (pp. 1–8). New York; ACM. doi:10.1145/2463372.2463378.

Bonyadi, M. R., & Michalewicz, Z. (2014). SPSO2011—analysis of stability, local convergence, and rotation sensitivity. In *Genetic and evolutionary computation conference* (pp. 9–15). Vancouver, Canada. ACM. doi:10.1145/2576768.2598263.

Bonyadi, M. R., Michalewicz, Z., & Li, X. (2014). An analysis of the velocity updating rule of the particle swarm optimization algorithm. *Journal of Heuristics*. doi:10.1007/s10732-014-9245-2.

Chen, D. B., & Zhao, C. X. (2009). Particle swarm optimization with adaptive population size and its application. *Applied Soft Computing*, *9*(1), 39–48. doi:10.1016/j.asoc.2008.03.001.

Cheng, M.-Y., Huang, K.-Y., & Chen, H.-M. (2011). Dynamic guiding particle swarm optimization with embedded chaotic search for solving multidimensional problems. *Optimization Letters*, *6*(6), 719–729. doi:10.1007/s11590-011-0297-z.

Clerc, M. (2006). *Particle swarm optimization*. Chichester: Wiley-ISTE.

Clerc, M., & Kennedy, J. (2002). The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73. doi:10.1109/4235.985692.

Deb, K., Joshi, D., & Anand, A. (2002). Real-coded evolutionary algorithms with parent-centric recombination. In *Congress on evolutionary computation* (pp. 61–66). Honolulu: IEEE. doi:10.1109/CEC.2002.1006210.

Engelbrecht, A. (2005). *Fundamentals of computational swarm intelligence*. Hoboken, NJ: Wiley.

Engelbrecht, A. (2011). Scalability of a heterogeneous particle swarm optimizer. In *Symposium on swarm intelligence* (pp. 1–8). Paris: IEEE. doi:10.1109/SIS.2011.5952563.

Engelbrecht, A. (2012). Particle swarm optimization: Velocity initialization. In *Congress on evolutionary computation* (pp. 1–8). Brisbane: IEEE.

García-Nieto, J., & Alba, E. (2011). Restart particle swarm optimization with velocity modulation: A scalability test. *Soft Computing*, *15*(13), 2221–2232. doi:10.1007/s00500-010-0648-1.

Ghosh, S., Das, S., Kundu, D., Suresh, K., Panigrahi, B. K., & Cui, Z. (2010). An inertia-adaptive particle swarm system with particle mobility factor for improved global optimization. *Neural Computing and Applications*, *21*(4), 237–250. doi:10.1007/s00521-010-0356-x.

Gut, A. (2009). *An intermediate course in probability*. New York: Springer.

Hansen, N., Ros, R., Mauny, N., Schoenauer, M., & Auger, A. (2011). Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing*, *11*(10), 5755–5769. doi:10.1016/j.asoc.2011.03.001.

Hao, G., & Wenbo, X. (2011). A new particle swarm algorithm and its globally convergent modifications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *41*(7), 1334–1351. doi:10.1109/tsmcb.2011.2144582.

Helwig, S., & Wanka, R. (2007). Particle swarm optimization in high-dimensional bounded search spaces. In *Swarm intelligence symposium* (pp. 198–205). Honolulu: IEEE. doi:10.1109/SIS.2007.368046.

Helwig, S., & Wanka, R. (2008). Theoretical analysis of initial particle swarm behavior. In *International conference on parallel problem solving from nature* (pp. 889–898). Berlin: Springer. doi:10.1007/978-3-540-87700-4_88.

Hsieh, S. T., Sun, T. Y., Liu, C. C., & Tsai, S. J. (2009). Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems Man and Cybernetics Part B: Cybernetics*, *39*(4), 444–456. doi:10.1109/Tsmcb.2008.2006628.

Huang, H., Qin, H., Hao, Z., & Lim, A. (2010). Example-based learning particle swarm optimization for continuous optimization. *Information Sciences*. doi:10.1016/j.ins.2010.10.018.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Murphy, K. (2010). Time-bounded sequential parameter optimization. In *Learning and intelligent optimization* (pp. 281–298). Berlin: Springer.

Jiang, M., Luo, Y., & Yang, S. (2007a). Particle swarm optimization-stochastic trajectory analysis and parameter selection. *Swarm intelligence focus on ant and particle swarm optimization*. Wien: I-TECH Education and Publishing.

Jiang, M., Luo, Y. P., & Yang, S. Y. (2007b). Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information Processing Letters*, *102*(1), 8–16. doi:10.1016/j.ipl.2006.10.005.

Kennedy, J. (2003). Bare bones particle swarms. In *Swarm intelligence symposium* (pp. 80–87). doi:10.1109/SIS.2003.1202251.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *International conference on neural networks* (Vol. 4, pp. 1942–1948). Piscataway: IEEE.

Lehre, P. K., & Witt, C. (2013). Finite first hitting time versus stochastic convergence in particle swarm optimisation. In L. Di Gaspero, A. Schaerf, & T. Stützle (Eds.), *Advances in metaheuristics*. New York: Springer.

Li, X., & Yao, X. (2011). Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, *16*(4), 210–224.

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*(5), 281–295. doi:10.1109/Tevc.2005.857610.

Malan, K., & Engelbrecht, A. P. (2008). Algorithm comparisons and the significance of population size. In *IEEE World Congress on computational intelligence* (pp. 914–920). Hong Kong: IEEE. doi:10.1109/CEC.2008.4630905.

Matyas, J. (1965). Random optimization. *Automation and Remote Control*, *26*(4), 246–253.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, *8*(5), 204–210. doi:10.1109/TEVC.2004.826074.

Montes de Oca, M. A., Aydın, D., & Stützle, T. (2011). An incremental particle swarm for large-scale continuous optimization problems: An example of tuning-in-the-loop (re) design of optimization algorithms. *Soft Computing*, *15*(13), 2233–2255. doi:10.1007/s00500-010-0649-0.

Montes de Oca, M. A., & Stützle, T. (2008). Convergence behavior of the fully informed particle swarm optimization algorithm. In *Genetic and evolutionary computation conference* (pp. 71–78). New York: ACM. doi:10.1145/1389095.1389106.

Montes de Oca, M. A., Stützle, T., Birattari, M., & Dorigo, M. (2009). Frankenstein's PSO: A composite particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, *13*(7), 1120–1132. doi:10.1109/Tevc.2009.2021465.

Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Application*, *2008*(5), 1–10. doi:10.1155/2008/685175.

Poli, R. (2009). Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. *IEEE Transactions on Evolutionary Computation*, *13*(6), 712–721. doi:10.1109/Tevc.2008.2011744.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence*, *1*(1), 33–57. doi:10.1007/s11721-007-0002-0.

Potter, M., & De Jong, K. (1994). *A cooperative coevolutionary approach to function optimization. Parallel problem solving from nature* (pp. 249–257). Berlin: Springer. doi:10.1007/3-540-58484-6_269.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, *8*(5), 240–255. doi:10.1109/tvec.2004.826071.

Rockafellar, R. T. (1996). *Convex analysis* (Vol. 28). Princeton: Princeton University Press.

Schmitt, M., & Wanka, R. (2013). Particle swarm optimization almost surely finds local optima. In *Genetic and evolutionary computation conference, Amsterdam, The Netherlands* (pp. 1629–1636). New York: ACM. doi:10.1145/2463372.2463563.

Shi, Y., & Eberhart, R. (1998a). A modified particle swarm optimizer. In *World Congress on computational intelligence* (pp. 69–73). Los Alamitos: IEEE. doi:10.1109/icec.1998.699146.

Shi, Y., & Eberhart, R. (1998b). Parameter selection in particle swarm optimization. In *Evolutionary programming VII* (pp. 591–600). Berlin: Springer. doi:10.1007/BFb0040810.

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

Spears, W. M., Green, D. T., & Spears, D. F. (2010). Biases in particle swarm optimization. *International Journal of Swarm Intelligence Research*, *1*(4), 34–57. doi:10.4018/jsir.2010040103.

Spiegel, M. R. (1959). *Theory and problems of vector analysis: Schaum's outline series*. New York: McGraw-Hill.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y., Auger, A., et al. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Report 2005005, Technical Report.

Tang, K., Yao, X., Suganthan, P. N., MacNish, C., Chen, Y. P., Chen, C. M., et al. (2007). Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. Nature Inspired Computation and Applications Laboratory, USTC, China, Technical Report.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, *85*(8), 317–325. doi:10.1016/S0020-0190(02)00447-7.

Tu, Z., & Lu, Y. (2004). A robust stochastic genetic algorithm (StGA) for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *8*(7), 456–470. doi:10.1109/TEVC.2004.831258.

Van den Bergh, F., & Engelbrecht, A. (2002). A new locally convergent particle swarm optimiser. In *Systems, man and cybernetics, Hammamet, Tunisia* (Vol. 3, pp. 96–101): Los Alamitos: IEEE.

Van den Bergh, F., & Engelbrecht, A. P. (2001). Effects of swarm size on cooperative particle swarm optimisers. In *Genetic and evolutionary computation conference, San Fransisco, USA*.

Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(5), 225–239. doi:10.1109/TEVC.2004.826069.

Van den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences*, *176*(10), 937–971. doi:10.1016/j.ins.2005.02.003.

Van den Bergh, F., & Engelbrecht, A. P. (2010). A convergence proof for the particle swarm optimiser. *Fundamenta Informaticae*, *105*(6), 341–374. doi:10.3233/FI-2010-370.

Vesterstrom, J., & Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Congress on evolutionary computation* (Vol. 2, Vol. 1982, pp. 1980–1987): Los Alamitos: IEEE. doi:10.1109/CEC.2004.1331139.

Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., & Tian, Q. (2011). Self-adaptive learning based particle swarm optimization. *Information Sciences*, *181*(20), 4515–4538. doi:10.1016/j.ins.2010.07.013.

Wilke, D. (2005). *Analysis of the particle swarm optimization algorithm*. Pretoria: University of Pretoria.

Wilke, D. N., Kok, S., & Groenwold, A. A. (2007a). Comparison of linear and classical velocity update rules in particle swarm optimization: Notes on diversity. *International Journal for Numerical Methods in Engineering*, *70*(10), 962–984. doi:10.1002/nme.1867.

Wilke, D. N., Kok, S., & Groenwold, A. A. (2007b). Comparison of linear and classical velocity update rules in particle swarm optimization: Notes on scale and frame invariance. *International Journal for Numerical Methods in Engineering*, *70*(10), 985–1008. doi:10.1002/nme.1914.

Witt, C. (2009). Why standard particle swarm optimisers elude a theoretical runtime analysis. In *Foundations of genetic algorithms, New York, NY, USA* (pp. 13–20). New York: ACM. doi:10.1145/1527125.1527128.

Xinchao, Z. (2010). A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing*, *10*(1), 119–124. doi:10.1016/j.asoc.2009.06.010.

Zhao, S., Liang, J., Suganthan, P., & Tasgetiren, M. (2008). Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In *IEEE World Congress on computational intelligence* (pp. 3845–3852). Los Alamitos: IEEE. doi:10.1109/CEC.2008.4631320.