2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Dimensions Cooperate by Euclidean Metric in Particle Swarm Optimization

Zezhou Li, Junqi Zhang†, Wei Wang and Jing Yao
Department of Computer Science and Technology
Key Laboratory of Embedded System and Service Computing, Ministry of Education
Tongji University, Shanghai, 200092, China
† Corresponding author (*E-mail*: zhangjunqi@tongji.edu.cn)

***Abstract*—Since Particle Swarm Optimization (PSO) was introduced, variants of PSO have usually updated velocities of particles in each dimension independently in the high-dimensional space. This paper proposes a Dimensionally Cooperative PSO (DCPSO), in which dimensions cooperate to update velocities of particles through Euclidean metric. The Euclidean metric first builds *pbest*-centered and *gbest*-centered hyperspheres. And then, velocity vectors of particles are derived from stochastic points obeying a distribution within the hyperspheres for dimensions cooperating. DCPSO investigates such cooperation of dimensions through Euclidean metric, instead of updating each dimension independently. Compared with the traditional PSO, DCPSO is validated by simulations on the 20 standard benchmark problems from CEC 2013. Furthermore, DCPSO shows more rotationally-invariant than the traditional PSO from the results. Additionally, the differences between the behaviors of the traditional PSO and the proposed DCPSO are analyzed from the aspect of the search space. Meanwhile, the curse of dimensionality is illustrated by comparisons between the traditional PSO and DCPSO in distinct dimensions.**

I. Introduction

Inspired by the social behaviors of birds flocking, Particle Swarm Optimization (PSO) is a swarm intelligence technique introduced by James Kennedy and Russell Eberhart in 1995 [1][2]. As an excellent algorithm for optimizing problems, PSO can be used in a wide range of applications [3].

In PSO, a particle records its own best position, called *pbest*, and the best postion of the whole swarm, called *gbest*. And in the next instant, the velocity of the particle contains three components: inertia, a *pbest*-guiding vector and a *gbest*-guiding vector. The last two components in the traditional PSO are usually formed by comparing *pbest* or *gbest* to the present position of the particle in each dimension independently. In other words, each dimension is independent of another, and there is no cooperation among dimensions when updating velocities in the traditional PSO. Because of mutual independence among dimensions, the search space of the last two components can be seen as *pbest*-centered and *gbest*-centered hypercuboids in the high-dimensional space. And the particle is located at one of vertices of each hypercuboid.

On the other hand, cooperation strategies related to dimensions have been extensively studied to effectively optimize PSO. Introduced by van den Bergh and Engelbrecht in 2000, Cooperative PSO (CPSO) achieves improvement by using multiple swarms to search subspaces for cooperation, so that it optimizes different components of the solution vector cooperatively [4][5]. Later, the effects of swarm size on CPSO are analyzed [6]. Another variant is Comprehensive Learning PSO (CLPSO), where each particle is able to learn not only its own *pbest*, but also the others' *pbest* for cooperation in each dimension [7]. Sequently, CLPSO has several improvements, such as the adaptive CLPSO (A-CLPSO) [8], CLPSO improved by generalised opposition-based learning [9], and so forth. It shows that CLPSO and its improvements have outperformances, especially in multimodel problems [7]. More similar to dimensional cooperation, a polar PSO has been studied through the conversion from the Cartesian space to the polar space [10]. However, CPSO and CLPSO are still updating velocities in each dimension independently, and after the conversion from the Cartesian space to the the polar space, the polar PSO also updates the velocities component-by-component.

Differing from the traditional PSO, this paper probes into the search space of PSO and proposes a Dimensionally Cooperative PSO (DCPSO), in which dimensions cooperate through Euclidean metric. DCPSO, first, builds *pbest*-centered and *gbest*-centered hyperspheres through Euclidean metric in the high-dimensional space. Then, the last two components of velocities in the next instant are derived from stochastic points obeying a distribution within the hyperspheres for dimensions cooperating.

The rest of this paper is organized as follows. In Section II, the traditional PSO is reviewed first. Then, DCPSO is presented and the relation and comparison with the traditional PSO are illuminated in Section III. To demonstrate the performance of DCPSO comparing to the traditional PSO, both algorithms are tested with a set of 20 standard benchmark problems from CEC 2013 in Section IV. At last, conclusions and the future work will be described in Section V.

II. The Traditional PSO

Mathematically, the traditional PSO can be formulated as a function as follows:

$$\begin{aligned} v_i^j(t+1) = w v_i^j(t) + c_1 r_1 (p_i^j(t) - x_i^j(t)) \\ + c_2 r_2 (g^j(t) - x_i^j(t)) \end{aligned} \tag{1}$$

$$x_i^j(t+1) = x_i^j + v_i^j(t+1) \tag{2}$$

where $v_i^j(t)$ is the velocity of particle i in dimension j at instant t; $x_i^j(t)$ is the position of particle i in dimension j at

instant t; p_i^j denotes the previously best position of particle i in dimension j; g^j refers to the previously best position of the swarm in dimension j; introduced by Shi and Eberhart in 1998, w is the inertia weight to improve the performance of PSO [11]; c_1 and c_2 are two acceleration constants, as two parameters in PSO; random real numbers r_1 and r_2 are chosen anew for each dimension.

Because of dimensional independence, in the traditional PSO, the search space of the *pbest*-guiding vector of a velocity is a *pbest*-centered hypercuboid, while the search space of the *gbest*-guiding vector of a velocity is a *gbest*-centered hypercuboid. Specifically, for a particle at some instant, it builds two hypercuboids and it is located at one of vertices of each hypercuboid. Then, a stochastic point is chosen within each hypercuboid to be the vector end for one of the last two components.

III. The Dimensionally Cooperative PSO (DCPSO)

A. DCPSO

The traditional PSO updates velocities in each dimension independently. In other words, when the traditional PSO updates velocities of particles in the high-dimensional space, each dimension utilizes only the presented dimension's information, without other dimensions' information.

In DCPSO, a particle updates its velocity with comprehensive information of all dimensions. This paper adopts Euclidean metric to build *pbest*-centered and *gbest*-centered hyperspheres for comprehending information of complete dimensions, instead of the hypercuboids resulted from independent updating in dimensions. The radii of hyperspheres are calculated as follows:

$$R_i^p = ||\mathbf{x}_i - \mathbf{p}_i|| = \sqrt{\sum_{j=1}^{n}(x_i^j - p_i^j)^2} \tag{3}$$

$$R_i^g = ||\mathbf{x}_i - \mathbf{g}|| = \sqrt{\sum_{j=1}^{n}(x_i^j - g^j)^2} \tag{4}$$

where R_i^p is the radius of *pbest*-centered hypersphere for particle i; R_i^g is the radius of *gbest*-centered hypersphere for particle i; $\mathbf{x}_i$ is the present position of the particle; $\mathbf{p}_i$ denotes the coordinate of *pbest*; while $\mathbf{g}$ refers to the coordinate of *gbest*; n is the number of dimensions.

Thus, the equations of the hyperspheres can be given as follows:

$$||\mathbf{x} - \mathbf{p}_i|| = \sqrt{\sum_{j=1}^{n}(x_j - p_i^j)^2} = R_i^p \tag{5}$$

$$||\mathbf{x} - \mathbf{g}|| = \sqrt{\sum_{j=1}^{n}(x_j - g^j)^2} = R_i^g \tag{6}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a variable in an n-dimensional space.

According to the above two formulae, particle i is located at the two hyperspheres. Then two stochastic points at the hyperspheres are chosen uniformly at random from the hyperspheres, denoting $\mathbf{x}_{sp}^p$ and $\mathbf{x}_{sp}^g$, respectively. The method for choosing a point uniformly at random from the hyperspheres can be described as **Algorithm 1** [12].

Algorithm 1 Generating a point uniformly at random from a hypersphere (in an n-dimensional space)

1: **BEGIN**
2: Generate an n-dimensional point, $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, of normal deviates $(x_i \sim N(0,1), 1 \le i \le n)$;
3: Calculate the L_2 norm of the point $\mathbf{x}$, that is, $r = ||\mathbf{x}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$;
4: The vector $\frac{1}{r}\mathbf{x}(r \neq 0)$ is uniformly distributed from a hypersphere;
5: **END**

And then, through two stochastic lengths between zero and the radii of the two hyperspheres, respectively, the two stochastic points can be shrunken within hyperspheres. Let the two points denote $\mathbf{x}^p$, which is derived from the *pbest*-centered hyperpsphere, and $\mathbf{x}^g$, which is derived from the *gbest*-centered hypersphere. Then, a vector from the position of particle i to $\mathbf{x}_i^p$ and a vector from the position i to $\mathbf{x}_i^g$ are built. Instead of the last two traditional components, those vectors become components of the next instant velocity of particle i. Hence, the formula for updating velocity is calculated as follows:

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1(\mathbf{x}_i^p - \mathbf{x}_i(t)) + c_2(\mathbf{x}_i^g(t) - \mathbf{x}_i(t)) \tag{7}$$

$$\mathbf{x}_i^p = r_1'(\mathbf{x}_{sp}^p - \mathbf{p}_i) + \mathbf{p}_i \tag{8}$$

$$\mathbf{x}_i^g = r_2'(\mathbf{x}_{sp}^g - \mathbf{g}) + \mathbf{g} \tag{9}$$

where $\mathbf{v}_i(t)$ is the velocity of the particle, which is a vector form of $v_i^j(t)(1 \le j \le n)$; $\mathbf{x}_{sp}^p$ denotes a stochastic point at *pbest*-centered sphere; $\mathbf{x}_{sp}^g$ refers to a stochastic point at *gbest*-centered sphere; r_1' and r_2' are two random real numbers between zero and one.

Actually, r_1' and r_2' are not necessary to distribute uniformly in $(0,1)$. This paper tries four distributions. And the following table lists (Table I) the four distributions with their probability density functions (pdf) and cumulative distribution functions (cdf).

TABLE I. Four distributions in the range of $(0,1)$

	pdf, $f(x) =$	cdf, $F(x) =$
1	1	x
2	$\frac{1}{2\sqrt{x}}$	$\sqrt{x}$
3	$\frac{1}{3\sqrt[3]{x^2}}$	$\sqrt[3]{x}$
4	$\frac{1}{4\sqrt[4]{x^3}}$	$\sqrt[4]{x}$

According to the above table (Table I), the random real number r' can be rewritten as r, r^2, r^3 and r^4, respectively, where r obeys uniform distribution in $(0,1)$. This paper conducts simulations to compare the performances not only between the traditional PSO and DCPSO, but also among the four kinds of DCPSO. From the simulation results in the next section, it seems that the third one, among the above four

distributions, performs well. Thus, the formulae of positions of two stochastic points within *pbest*-centered and *gbest*-centered hyperspheres can be rewritten as follows:

$$\mathbf{x}_i^p = r_1^m(\mathbf{x}_{sp}^p - \mathbf{p}_i) + \mathbf{p}_i \tag{10}$$

$$\mathbf{x}_i^g = r_2^m(\mathbf{x}_{sp}^g - \mathbf{g}) + \mathbf{g} \tag{11}$$

where r_1 and r_2 is uniformly distributed in $(0, 1)$; $m \leq 4$ is a positive integer.

DCPSO can be elaborated in **Algorithm 2**.

Algorithm 2 DCPSO

BEGIN
Initialize velocities and positions of the particle swarm;
Update *pbest* of each particle and *gbest* of the swarm;
Set $FEs = 0$;
while $FEs < MaxFEs$ **do**
  for each particle in the swarm **do**
    Calculate R_i^p and R_i^g according to (3) and (4);
    According to **Algorithm 1**, choose a point in each hypersphere of (5) or (6) uniformly;
    Choose a stochastic point obeying a distribution within each hypersphere according to (8) and (9);
    Update the velocity of each particle according to (7);
    Update the position of each particle according to (2);
    Calculate the fitness of each particle;
  end for
  if the fitness of *pbest* is improved **then**
    Update *pbest* and the fitness value of *pbest*;
  end if
  if the fitness of *gbest* is improved **then**
    Update *gbest* and the fitness value of *gbest*;
  end if
  $FEs \leftarrow FEs + Particle\_Number$;
end while
END

B. The relation between the traditional PSO and DCPSO

In order to compare the traditional PSO's and DCPSO's potential search spaces, this paper temporarily omits the first component of a velocity, inertia, i.e. $w\mathbf{v}_i$. Hence, the search space is the composition of *pbest*-centered and *gbest*-centered spaces. Due to the similarity between *pbest*-centered and *gbest*-centered spaces, this paper uses "*best*-centered" to refer to *pbest*-centered or *gbest*-centered. Thus, the search space is probed into by comparison between the *best*-centered spaces in the traditional PSO and DCPSO. For convenience, let both of parameters c_1 and c_2 equate 2. Hence, *best*-centered spaces in the traditional PSO and DCPSO are a hypercuboid and a hypersphere, respectively. Fig. 1 shows the comparison of *best*-centered spaces between the traditional PSO and DCPSO in a 3-dimension space.

Here, this paper defines two concepts: the Reachable Volume (RV) and the Expected Volume (EV) of a *best*-centered space.

Definition 1: Let p denote a stochastic point, only able to reach every point in a space point set S, i.e. every point in

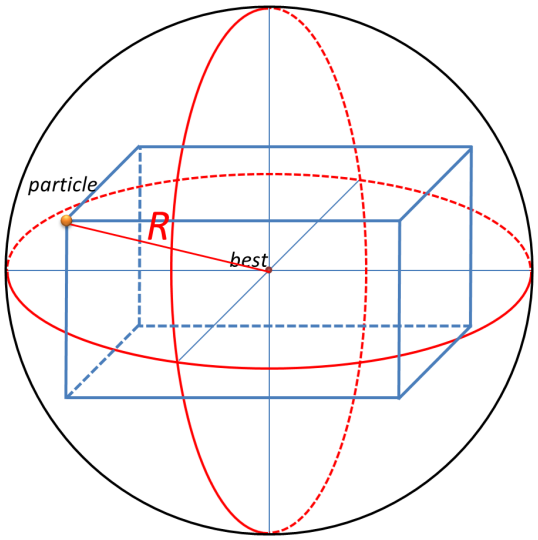


Fig. 1. *best*-centered spaces of the traditional PSO (a cuboid) and DCPSO (a sphere) in a 3-D space (where R is R_i^p or R_i^g)

S is reachable by p and meanwhile points not belonging to S are not reachable by p. The Reachable Volume (RV) of S is defined as the volume of the space.

Example 1: Let S be a ball with the radius R_S, regardless of the probability density of p in the ball, RV of S is the volume (V_S) of the space, i.e. $V_S = \frac{4}{3}\pi R_S^3$.

Definition 2: Based on **Difination 1**, let p be with a probability density in S. Let S' denote a set, in which every element is a space point set built by p with its probability density in S. The Expected Volume (EV) of S is the expectation of RV of the space in S'.

Based on *Example 1*, here are two extra examples.

Example 2: Suppose that p is uniformly distributed in every point in the ball S. The cumulative distribution function (cdf) is $F(r) = \frac{\frac{4}{3}\pi r^3}{\frac{4}{3}\pi R_S^3} = \frac{r^3}{R_S^3}$. R'_S, the expected R_S, is $\int_0^{R_S} r dF(r) = \frac{3}{4}R_S$. That is to say, expected positions of stochastic points in each direction vector build a new sphere with three quarters of radius of the original sphere. Hence, EV of the original ball is $V'_S = (\frac{3}{4})^3 V_S = \frac{9}{16}\pi R_S^3$.

Example 3: If the probability of each direction vector from the center of the ball is mutually equal and stochastic points are uniformly distributed in every direction vector, R'_S, the expected R_S, is half of R_S. That is to say, expected positions of stochastic points in each direction vector build a new sphere with half radius of the original sphere. Hence, EV of the original sphere is $V'_S = \frac{1}{2^3}V_S = \frac{1}{6}\pi R_S^3$.

Through **Definition 1**, RV of the *best*-centered spaces of the traditional PSO and DCPSO can be calculated as follows [13]:

$$V_R^t = \prod_{j=1}^{n} a_j \tag{12}$$

$$V_R^{DC} = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma(\frac{n}{2}+1)} \tag{13}$$

where V_R^t denotes RV of the search space of the traditional PSO, while V_R^{DC} refers to RV of the search space of DCPSO; a_j is the side length of the *best*-centered hypercuboid of the traditional PSO in dimension j; n is the number of dimensions; R is the radius of the *best*-centered hypersphere of DCPSO.

In a *best*-centered hypercuboid of the traditional PSO, the expected side length of the new hypercuboid is half of the original side length in each dimension. Hence, EV of *best*-centered space of the traditional PSO can be calculated as follows:

$$V_E^t = \frac{1}{2^n} \prod_{j=1}^{n} a_j \tag{14}$$

Similar to *Example 3*, EV of *best*-centered space of DCPSO with the first distribution defined in Table I, which is uniformly distributed in each direction (thus, the expected radius will be half of the original radius according to *Example 3*), can be calculated as follows:

$$V_E^{DC} = \frac{\pi^{\frac{n}{2}} R^n}{2^n \Gamma(\frac{n}{2}+1)} \tag{15}$$

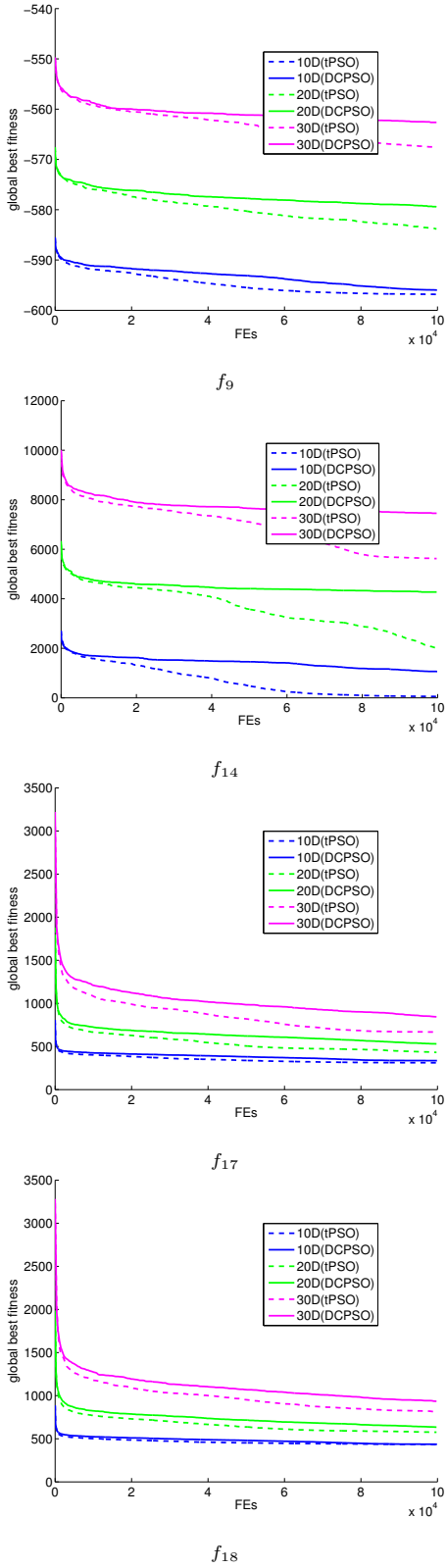
Similarly, EV of the *best*-centered space of DCPSO with the above distributions are as follows (Table II).

TABLE II. EV OF *best*-CENTERED SPACE OF DCPSO

	EV of *best*-centered space
1	$\frac{\pi^{\frac{n}{2}} R^n}{2^n \Gamma(\frac{n}{2}+1)}$
2	$\frac{\pi^{\frac{n}{2}} R^n}{3^n \Gamma(\frac{n}{2}+1)}$
3	$\frac{\pi^{\frac{n}{2}} R^n}{4^n \Gamma(\frac{n}{2}+1)}$
4	$\frac{\pi^{\frac{n}{2}} R^n}{5^n \Gamma(\frac{n}{2}+1)}$

Obviously, RV of *best*-centered space of DCPSO is strictly larger than that of the traditional PSO and EV of *best*-centered space of DCPSO. And RV of DCPSO with the first distribution defined in Table I is also strictly larger than that of the traditional PSO. Although enlargement is able to search more space, it brings the curse of dimensionality. Hence, for relieving the problem brought by enlargement, a suitable distribution is needed to be imposed to control EV of the search space.

In conclusion, the traditional PSO builds two hypercuboids, a *pbest*-centered hypercuboid and a *gbest*-centered hypercuboid, while DCPSO builds two hyperspheres, a *pbest*-centered hypersphere and a *gbest*-centered hypersphere. Meanwhile, a particle is located not only at a vertex of hypercuboids, but also at the hyperspheres. Hence, the hypercuboids are always inscribed in the hyperspheres. In terms of RV, the reachable volume of the *best*-centered hypersphere is larger than the reachable volume of the *best*-centered hypercuboid. What is more, the search space is the superposition of *pbest*-centered space and *gbest*-centered space. Hence, the search space of DCPSO is larger than the traditional PSO. However,



f_9 f_{14} f_{17} f_{18}

Fig. 2. The curse of dimensionality

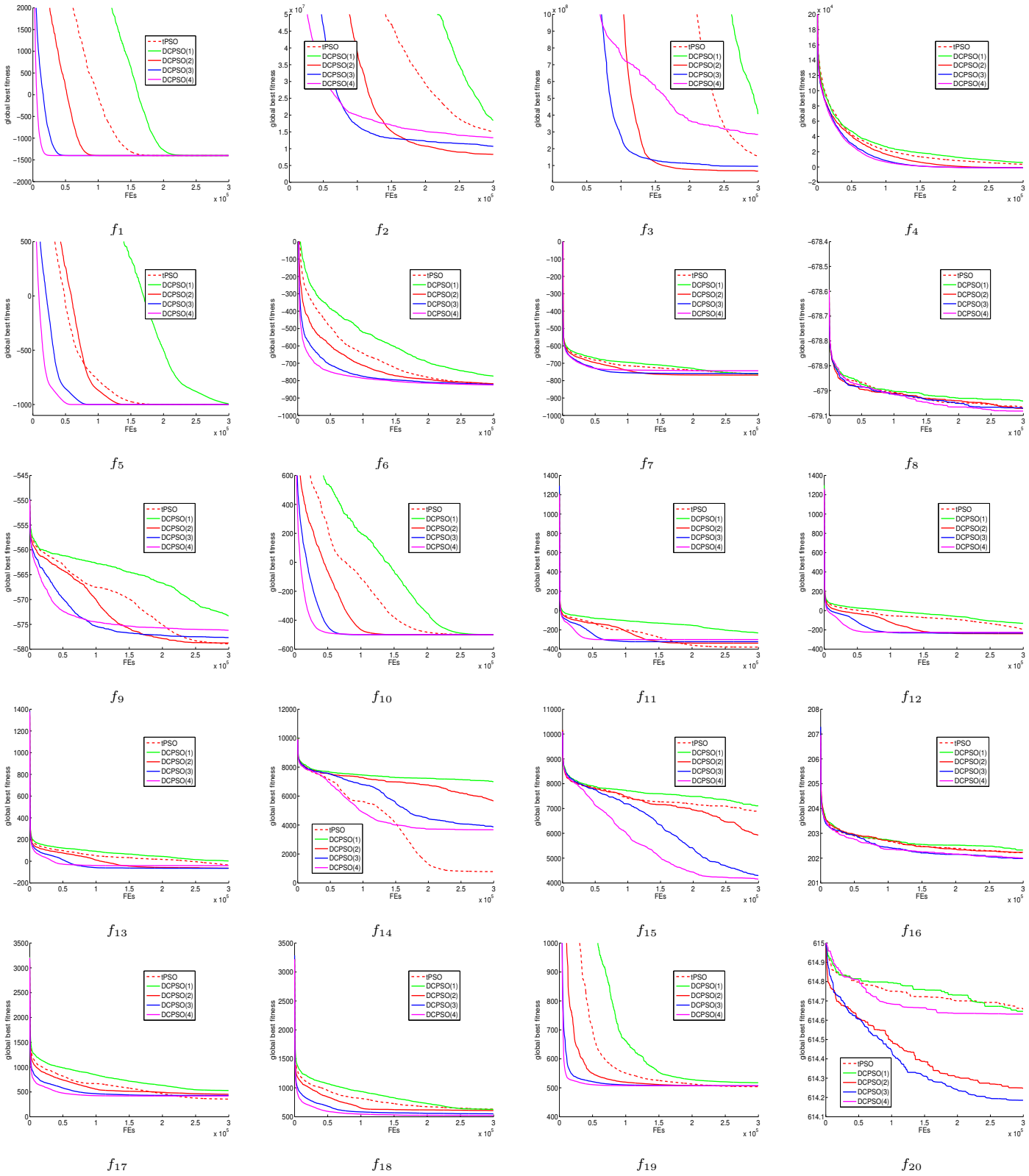


Fig. 3. The comparison of average convergence curves over 51 runs on 20 benchmark functions

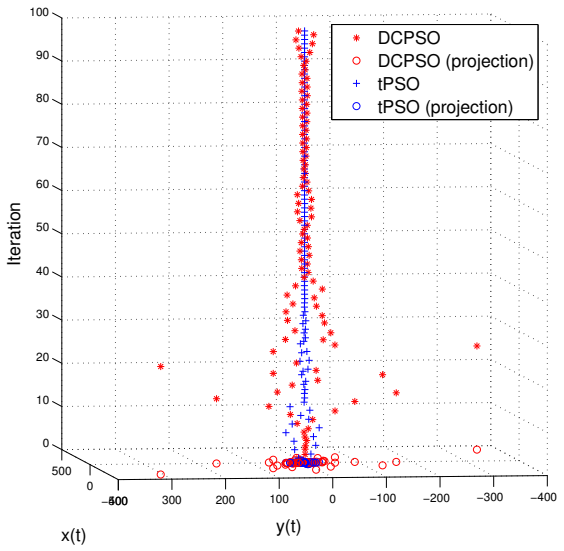


Fig. 4. The convergence of the traditional PSO and DCPSO(3) ($w = 0, \mathbf{x}(0) = (10, 10), \mathbf{v}(0) = (10, 10), c_1 = c_2 = 2$)

with the increment of the number of dimensions, the search space may be too large to converge. To relieve the curse of dimensionality, a proper distribution is imposed within the *best*-centered hypersphere. And in terms of EV, with some certain distribution, the expected volume of the *best*-centered hypersphere is not necessarily larger than that of the *best*-centered hypercuboid.

IV. Simulation Results

It is necessary to have simulations of DCPSO on some benchmark problems for taking a glance at the performance of it. However, because of the curse of dimensionality, DCPSO with the first distribution defined in Table I can hardly show its advantages over the traditional PSO. In Fig. 2, the performances of the traditional PSO (tPSO, with w linearly decreasing from 0.9 to 0.4 and $c_1 = c_2 = 2$) and DCPSO (with w linearly decreasing from 0.9 to 0.4 and the first distribution defined in Table I) are displayed under four benchmark problems from CEC 2013 (information of four benchmark problems included in Table III [14]). From Fig. 2, it is shown that, with the increment of the number of dimensions, the performance of DCPSO with the first distribution defined in Table I may get worse. That is because the curse of dimensionality leads to a too large search space for DCPSO in terms of RV. Hence, this paper compares the performances of the traditional PSO and DCPSO with four distinct distributions defined in Table I and it seems that DCPSO with the third distribution performs well. They are run on the 20 standard benchmark problems from CEC 2013. Table III shows the information of the 20 tested benchmark problems and Table IV shows the information of the 5 tested PSO, which are tPSO (the traditional PSO) and DCPSO(k) (DCPSO with the kth distribution defined in Table I, $k \leq 4$ is a positive integer).

One criterion applied to terminate simulation of the algorithms is function evaluations (FEs) reaching maximum number of function evaluations (MaxFEs), which has already implied in **Algorithm 2**. This paper sets MaxFEs as $10000D = 300000$. 51 independent stimulation runs are conducted for each algorithm. Fig. 3 reveals the simulation results, and it seems that DCPSO(3) performs well. To compare DCPSO(3) with the traditional PSO in the aspect of convergence, a simulation tested, of which the results can be seen in Fig. 4. Eliminating the impact of inertia, the simulation is under the following conditions: $w = 0, \mathbf{x}(0) = (10, 10), \mathbf{v}(0) = (10, 10), c_1 = c_2 = 2$. It can be seen that, obviously, the range of the search space of DCPSO(3) is larger than that of the traditional PSO. However, it can hardly guarantee the convergence of DCPSO(3). Further, for each algorithm, the average (mean) fitness value (Mean) and the standard deviation (Std. Dev.) over the 51 runs are calculated, and then algorithms are compared through the two values. The experimental results are shown in Table V.

TABLE III. Information of the 20 tested benchmark problems with the number of dimensions $D = 30$

No.	Type	Search Ranges
$f_1 - f_5$	Unimodal Functions	$[-100, 100]^D$
$f_6 - f_{20}$	Basic Multimodal Functions	$[-100, 100]^D$

TABLE IV. Parameters of the traditional PSO (tPSO) and DCPSO in simulation

Algorithms	Parameters w, c_1 and c_2
tPSO	w linearly decreases from 0.9 to 0.4, $c_1 = c_2 = 2$
DCPSO(k)	w linearly decreases from 0.9 to 0.4, without c_1 and c_2

From Fig. 3, it seems that DCPSO(3) performs well, and from the computational results presented in Table V, it is observed that DCPSO(3) shows superior performances over the traditional PSO for 11 out of the 20 benchmark problems: f_2, f_3, f_4, f_6, f_{10}, f_{12}, f_{13}, f_{15}, f_{16}, f_{18} and f_{20}. And in the two benchmark problems, f_1 and f_5, both of the traditional PSO and DCPSO(3) reach the optimum values. DCPSO(3) is able to have such performances because RV of the search space is larger than the traditional PSO. And meanwhile, it has a distribution to control its EV to relieve the problem, for the huge RV leads to the curse of dimensionality. However, the traditional PSO shows superior performances over the traditional PSO for 7 out of the 20 benchmark problems: f_7, f_8, f_9, f_{11}, f_{14}, f_{17} and f_{19}. Overall, each PSO has its merits.

TABLE VI. Rotation information of the two function pairs (f_{14}, f_{15}) and (f_{17}, f_{18})

No.	Function	f_i^*
f_{14}	Schwefel's Function	-100
f_{15}	Rotated Schwefel's Function	100
f_{17}	Lunacek Bi_Rastrigin Function	300
f_{18}	Rotated Lunacek Bi_Rastrigin Function	400

Notice the results between f_{14} and f_{15} and between f_{17} and f_{18}. The information of those four benchmark problems is shown in Table VI.

A notable result is that the performances of the traditional PSO in each of those two function pairs differ much more greatly than DCPSO. From Fig. 3, it shows that the curve graphs of the traditional PSO in each of those two function pairs differ greatly, while the curve graphs of DCPSO differ slightly. And from Table V, if f_i^*, which is a real constant to adjust the minimum values of each standard benchmark problem f_i from CEC 2013, is eliminated, the mean function

TABLE V. MEAN FUNCTION VALUES AND STANDARD DEVIATIONS OVER 51 RUNS OF THE 20 TESTED BENCHMARK PROBLEMS

Func. No.		tPSO	DCPSO(1)	DCPSO(2)	DCPSO(3)	DCPSO(4)
f_1	Mean	**-1.4000e+003**	**-1.4000e+003**	**-1.4000e+003**	**-1.4000e+003**	**-1.4000e+003**
f_1	Std. Dev.	5.6869e-026	2.8951e-026	**2.6883e-026**	3.7223e-026	3.9291e-026
f_2	Mean	1.4994e+007	1.8395e+007	**8.3045e+006**	1.0688e+007	1.3256e+007
f_2	Std. Dev.	1.0198e+014	1.3563e+014	5.3831e+013	7.0169e+013	**5.2601e+013**
f_3	Mean	1.5433e+008	4.0484e+008	**6.6681e+007**	9.5335e+007	2.8473e+008
f_3	Std. Dev.	4.2203e+016	1.7672e+018	2.4883e+016	**1.9835e+016**	6.1208e+017
f_4	Mean	3.5386e+003	5.9017e+003	-9.2595e+002	-1.0477e+003	**-1.0868e+003**
f_4	Std. Dev.	1.8992e+006	2.6048e+006	3.8270e+003	9.8272e+002	**8.0359e+001**
f_5	Mean	**-1.0000e+003**	-9.9330e+002	**-1.0000e+003**	**-1.0000e+003**	**-1.0000e+003**
f_5	Std. Dev.	**1.1348e-025**	1.3887e+002	3.0109e-009	3.7283e-009	1.8880e-009
f_6	Mean	-8.1878e+002	-7.7424e+002	-8.1792e+002	-8.2195e+002	**-8.2476e+002**
f_6	Std. Dev.	1.2899e+003	**1.2252e+003**	1.4579e+003	1.4555e+003	1.6829e+003
f_7	Mean	-7.6135e+002	-7.6159e+002	**-7.6807e+002**	-7.5866e+002	-7.4313e+002
f_7	Std. Dev.	1.9516e+002	1.1188e+002	**9.1813e+001**	2.3504e+002	4.5121e+002
f_8	Mean	-6.7907e+002	-6.7904e+002	-6.7907e+002	-6.7907e+002	**-6.7908e+002**
f_8	Std. Dev.	3.0651e-003	**1.0170e-003**	4.8644e-003	2.8533e-003	5.8100e-003
f_9	Mean	**-5.7895e+002**	-5.7336e+002	-5.7875e+002	-5.7766e+002	-5.7619e+002
f_9	Std. Dev.	1.8059e+001	2.8631e+001	**9.2059e+000**	9.8594e+000	1.2958e+001
f_{10}	Mean	-4.9963e+002	-4.9963e+002	**-4.9982e+002**	-4.9979e+002	**-4.9982e+002**
f_{10}	Std. Dev.	1.4969e-001	1.0547e-001	**8.9900e-003**	2.0840e-002	1.1841e-002
f_{11}	Mean	**-3.7862e+002**	-2.3120e+002	-3.3742e+002	-3.2243e+002	-3.0013e+002
f_{11}	Std. Dev.	**4.5716e+001**	2.2284e+003	3.0958e+002	5.0324e+002	1.2522e+003
f_{12}	Mean	-1.9552e+002	-1.3243e+002	**-2.3976e+002**	-2.3369e+002	-2.2491e+002
f_{12}	Std. Dev.	2.3582e+003	2.2257e+003	**4.2453e+002**	5.5673e+002	7.8376e+002
f_{13}	Mean	-3.3876e+001	2.3631e+000	-6.3588e+001	**-6.4544e+001**	-4.0152e+001
f_{13}	Std. Dev.	1.3434e+003	**5.2100e+002**	1.0099e+003	1.0039e+003	1.4525e+003
f_{14}	Mean	**7.8565e+002**	7.0010e+003	5.6623e+003	3.8780e+003	3.6756e+003
f_{14}	Std. Dev.	**7.9358e+004**	3.5293e+005	2.0398e+006	1.4670e+006	7.3548e+005
f_{15}	Mean	6.8710e+003	7.1084e+003	5.9306e+003	4.2877e+003	**4.1563e+003**
f_{15}	Std. Dev.	4.9695e+005	**2.2987e+005**	1.9329e+006	1.2534e+006	9.0453e+005
f_{16}	Mean	2.0222e+002	2.0233e+002	2.0223e+002	**2.0199e+002**	**2.0199e+002**
f_{16}	Std. Dev.	**1.2841e-001**	1.9478e-001	2.0448e-001	2.6777e-001	1.9639e-001
f_{17}	Mean	**3.5568e+002**	5.2893e+002	4.5289e+002	4.2932e+002	4.1771e+002
f_{17}	Std. Dev.	**4.3644e+002**	1.3215e+003	2.8272e+003	1.6481e+003	1.2026e+003
f_{18}	Mean	6.3208e+002	6.2707e+002	6.0713e+002	5.5161e+002	**5.2191e+002**
f_{18}	Std. Dev.	1.2113e+003	**8.0284e+002**	1.6146e+003	2.8470e+003	1.1339e+003
f_{19}	Mean	**5.0398e+002**	5.1751e+002	5.0664e+002	5.0696e+002	5.0689e+002
f_{19}	Std. Dev.	**1.8437e+000**	1.2147e+001	1.0613e+001	5.1581e+000	4.9766e+000
f_{20}	Mean	6.1466e+002	6.1465e+002	6.1425e+002	**6.1419e+002**	6.1463e+002
f_{20}	Std. Dev.	9.1348e-001	**8.3615e-001**	1.9767e+000	2.8666e+000	1.1945e+000

values of DCPSO in each of those two function pairs are very close, while the mean function values of the traditional PSO are not. The outcomes may be resulted from the concentration of particles parallel to the coordinate axes in the traditional PSO [15], while in DCPSO, the concentration is eliminated. For not exploring the search space on a component-by-component basis due to the concentration of particles parallel to the coordinate axes, the Incremental PSO with Local Search (IPSOLS) add a new particle to the population every k iterations [16]–[18]. However, both of the initialization rule applied to a new particle and velocities updating are independent in each dimension.

V. CONCLUSIONS

In this paper, we proposed a Dimensionally Cooperative PSO (DCPSO), based on dimensional cooperation through Euclidean metric. It is essentially novel because DCPSO is to propose such cooperation of dimensions through Euclidean metric, instead of updating each dimension independently.

This paper defines two concepts, the Reachable Volume (RV) and the Expected Volume (EV) of a space. Though Euclidean metric, it is obvious that EV for the search space of DCPSO will be larger than the traditional PSO. However, with the increment of the number of dimensions, RV of search space may be too large to converge, which is one of aspects of the curse of dimensionality. To relieve the problems in the high-dimensional space, this paper imposes four distributions on DCPSO to control DCPSO's EV. Thus, DCPSO is able to accelerate its convergence.

DCPSO is validated by comparison with the traditional PSO under 20 standard benchmark problems from CEC 2013 with the number of dimensions $D = 30$ over 51 runs. It seems that, among the four distributions, DCPSO with the third distribution performs well, which has superior performances over the traditional PSO for 11 out of 20 benchmark problems and in another two benchmark problems, both the traditional PSO and DCPSO reach the optimum values. It also shows that in rotated and non-rotated functions, the traditional PSO performs much differently while DCPSO differs slightly.

In our future work, we will try more metrics to study the performance of DCPSO, and further, we will consider the convergence condition of DCPSO. Meanwhile, we will test the performances of the traditional PSO and DCPSO in more non-rotated and rotated function pairs, in order to study the impact of rotation to the traditional PSO and DCPSO. What is more, Partially Dimensionally Cooperative PSO (PDCPSO), which is dimensionally cooperative in subspaces so as to further relieve the curse of dimensionality is also of our interests.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (NSFC), under Grants No. 61272271, 61332008, 61174158, 61103072, 61103071 and 61103068.

This program is also supported by the National Basic Research Program of China (973 Program) under Grant No. 2014CB340404, Natural Science Foundation Program of Shanghai under Grant No. 12ZR1434000, Fundamental Research Funds for the Central Universities under Grants No. 0800219201, Research Fund for the Doctoral Program of Higher Education of China (20110072120065) and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks (Perth, Australia)*, IEEE Service Center, Piscataway, NJ, IV, pp. 1942-1948, 1995.

[2] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," *Proceedings of the 6th International Symposium on Micromachine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.

[3] R. Poli, "Analysis of the Publications on the Applications of Particle Swarm Optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, no. 3, pp. 1-10, Jan. 2008.

[4] F. van den Bergh and A. P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," *South African Comput. J.*, vol. 26, pp. 84-90, Nov. 2000.

[5] F. van den Bergh and A. P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization," *South African Comput. J.*, vol. 8, no. 3, pp. 225-239, Jun. 2004.

[6] F. van den Bergh, A. P. Engelbrecht, "Effects of swarm size on cooperative particle swarm optimisers," *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, USA, 2001.

[7] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *Transactions on Evolutionary Computation*, vol. 10, no. 3, 2006.

[8] H. Wu, J. Geng, R. Jin, J. Qiu, W. Liu, J. Chen and S. Liu, "An improved comprehensive learning particle swarm optimization and its application to the semiautomatic design of antennas," *Antennas and Propagation, IEEE Transactions on*, vol 57, no 10, pp. 3018-3028, 2009.

[9] W. Wang, H. Wang and S. Rahnamayan, "Improving comprehensive learning particle swarm optimiser using generalised opposition-based learning," *International Journal of Modelling, Identification and Control*, vol. 14, no. 4, pp. 310-316, 2011.

[10] A. P. Engelbrecht and W. Matthysen, "A polar coordinate particle swarm optimiser," *Applied Soft Computing*, vol. 11, no. 1, pp. 1322-1339, 2011.

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," *Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence*, IEEE, pp. 69-73, 1998.

[12] G. Marsaglia. "Choosing a point from the surface of a sphere," *The Annals of Mathematical Statistics*, vol. 43, no. 2, pp. 645-646, 1972.

[13] G. Huber, "Gamma function derivation of n-sphere volumes," *Amer. Math. Monthly*, vol. 89, no. 5, pp. 301-302, 1982.

[14] J. J. Liang, B. Y. Qu, P. N. Suganthan and A. G. Hernández-Díaz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, and Technical Report, Nanyang Technological University, Singapore, Jan. 2013.

[15] W. M. Spears, D. T. Green and D. F. Spears, "Biases in particle swarm optimization," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 1, no. 2, pp. 34-57, 2010.

[16] M. A. M. de Oca, K. van den Enden and T. Stützle, "Incremental particle swarm-guided local search for continuous optimization," *Proceedings of the international workshop on hybrid metaheuristics (HM 2008)*, Springer, Heidelberg, pp. 72-86, 2008.

[17] M. A. M. de Oca, T. Stützle, K. van den Enden and M. Dorigo, "Incremental social learning in particle swarms," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 2, pp. 368-384, 2011.

[18] M. A. M. de Oca, D. Aydın and T. Stützle, "An incremental particle swarm for large-scale continuous optimization problems: an example of tuning-in-the-loop (re)design of optimization algorithms," *Soft Computing*, vol. 15, no. 11, pp. 2233-2255, 2011.