PLOS | ONE

RESEARCH ARTICLE

Lagrange Interpolation Learning Particle Swarm Optimization

Zhang Kai¤*☯, Song Jinchun☯, Ni Ke☯, Li Song☯

Mechanical Engineering and Automation, Northeast University, Shenyang, China

☯ These authors contributed equally to this work.
¤ Current Address: mailbox 319 of hydraulic pneumatic institute of northeast university, Shenyang city, China
* zk1210071@stumail.neu.edu.cn

Abstract

In recent years, comprehensive learning particle swarm optimization (CLPSO) has attracted the attention of many scholars for using in solving multimodal problems, as it is excellent in preserving the particles' diversity and thus preventing premature convergence. However, CLPSO exhibits low solution accuracy. Aiming to address this issue, we proposed a novel algorithm called LILPSO. First, this algorithm introduced a Lagrange interpolation method to perform a local search for the global best point (gbest). Second, to gain a better exemplar, one gbest, another two particle's historical best points (pbest) are chosen to perform Lagrange interpolation, then to gain a new exemplar, which replaces the CLPSO's comparison method. The numerical experiments conducted on various functions demonstrate the superiority of this algorithm, and the two methods are proven to be efficient for accelerating the convergence without leading the particle to premature convergence.

Citation: Kai Z, Jinchun S, Ke N, Song L (2016) Lagrange Interpolation Learning Particle Swarm Optimization. PLoS ONE 11(4): e0154191. doi:10.1371/journal.pone.0154191

Editor: Wen-Bo Du, Beihang University, CHINA

Received: December 22, 2015

Accepted: April 8, 2016

Published: April 28, 2016

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: The authors have no support or funding to report.

Competing Interests: The authors have declared that no competing interests exist.

1. Introduction

In 1995, Kennedy and Elberhart, inspired by the foraging behaviour of birds, proposed the particle swarm optimization (PSO) algorithm [1], which has attracted attention in the academic circles and has demonstrated its superiority in solving practical problems. PSO is a type of evolutionary algorithm, which is similar to the simulated annealing (SA) [2] algorithm. PSO starts from a random solution, searches for the optimal solution in iterative manner, and then evaluates the quality of the solution based on the fitness. PSO follows the current optimal value to find the global optimum, so it is simpler than the genetic algorithm (GA) [3], without the need for the "cross" and "mutation" operations. However, PSO suffers two problems: premature and slow convergence at the late stage. In the past two decades, many researchers have focused on addressing these problems by introducing some methods and concepts. A concept of inertia weight was introduced and applied in the formula by Y.Shi and Eberhart, they set the weight from 0.9 to 0.4 to provide a balance between exploitation and exploration [4][5]; On the basis of that, later researchers developed adaptive inertia weight and coefficients [6][7]. To avoid the premature convergence of the particle swarm, Riget J and Vesterstrim Js proposed a concept of

diversity: they set a lower bound of diversity to ensure the swarm has a good search ability [8]. A simulated annealing (SAPSO) idea was introduced to help a particle jump out of the local [9]. A grey relational analysis was introduced for changing the parameters of PSO to help improve the algorithm performance in [10]. A chaotic search idea to global search was proposed in [11], which was improved by introducing the sequence quadratic program (SQP) algorithm to accelerate convergence in [12]. The gradient search for accurate computation of the global minimum was proposed in [13]. Meanwhile, Some researchers focus on structural and heterogeneous factor, such as SIPSO [14], SFPSO [15] and LIPSO [16].

To solve the multimodal problems, J.J. Liang proposed the comprehensive learning particle swarm optimizer (CLPSO) [17]. Later, Liang and Suganthan [18] proposed an adaptive CLPSO with historical learning, for which the particles' learning probabilities are adjusted adaptively. On the basis of that finding, researchers realised that CLPSO's search method was quite efficient for finding the global optimum. However, CLPSO suffers a slow resolution. Aiming to this point, some improved CLPSO was proposed. An orthogonal experimental design was introduced in CLPSO to determine the best combination of learning from a particle's personal best position or its neighbourhood's historical best position [19]. Zheng et al. [20] proposed ACLPSO which adaptively sets the factors of the algorithm, i.e. the inertia weight and acceleration coefficient. Nasir [21] proposed a DNLPSO which used a learning strategy, whereby all other particles' historical best information was used to update a particle's velocity, as in CLPSO. However, in contrast to CLPSO, the exemplar particle was selected from a neighbourhood. The neighbourhoods were made dynamically in nature, i.e, they are reformed after certain intervals. Xiang Yu [22] introduced a kind of perturbation in to the iterative forms of the CLPSO. It determined the particles' learning probabilities between it's historical best values and the dimensional bounds. Because there is a bound for the perturbation, it will speed up the convergence. On the base of CLPSO, some multi-objective optimization problems can be solved by using Pareto dominance concept [23][24][25].

In this research, we proposed a novel improved algorithm, called LILPSO, which was based on CLPSO by introducing the Lagrange interpolation method. There are two main differences between LILPSO and CLPSO. First, When this algorithm performs as CLPSO's search method for some times, it will introduce one Lagrange interpolation computation for each dimension of the best point (gbest). This is a local search method, and it will help accelerating the convergence. Second, CLPSO selects the better one of the other two particles' historical optimum (pbest) as the exemplar at the d^{th} dimension. Compared with CLPSO, LILPSO selects three points, which are the i^{th} particles' historical optimum, another random particles' historical optimum and the global optimum (*gbest*), to perform the Lagrange interpolation, and then to obtain a parabola, whose optimum is the exemplar we expected.

The remainder of this article is organized as follows: Section 2 provides related works regarding PSO and CLPSO, and discusses the Lagrange interpolation theory. In Section 3, the proposed LILPSO is discussed in sufficient detail. Section 4 provides the experimental results on different functions to prove the superiority of LILPSO. Section 5 presents the paper's conclusions.

2. Related Works

2.1 PSO algorithm

Assuming the optimization problem is

$$\min f(x) = f(x_1, x_2 \cdots, x_n)$$

$$s.t. \quad x_i \in [L_i, U_i] \quad i = 1, 2, \cdots n$$

If the particle is denoted as $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$, Then the best position it experienced (the best fitness value) is $P_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$, also denoted by p_{best}, The index of the best position experienced in particle group represented by symbol g is denoted by Pg, or g_{best}. The speed of particle i is denoted by $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$, for each generation, its d^{th} dimension iteration functions are:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (p_{id}(t) - x_{id}(t)) + c_2 r_2 (p_{gd}(t) - x_{id}(t)) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

Where, c_1 and c_2 are all positive constants, called learning factors; r_1 and r_2 are random numbers, which are from 0 to 1; v_{id} is the -dimension speed of each particle. The first part of the right of the equal sign in Eq 1 is caused by the particle previous velocity, is called "inertia" part. The second part is "cognition" part, which illustrates that the particle thinks itself, as well as influences the particle information itself of the next step. The third part is the "social" part, which illustrates the information shared and mutual cooperation, as well as the influences on the swarm information of the next step.

2.2 CLPSO algorithm

CLPSO iteration function is different from the standard PSO.

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1 (p_{id}(t)' - x_{id}(t)) \quad (3)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (4)$$

where, $\omega = \omega_{max} - \frac{t}{max\_gen}(\omega_{max} - \omega_{min})$, $\omega_{max} = 0.9$ and $\omega_{min} = 0.4$. $p_{id}(t)'$ is the exemplar of the i^{th} particle in the d^{th} dimension. If the i^{th} particle does not update its historical optimum (*pbest*) continuously and over the gap m (usually m = 7), then a random number from 0 to 1 will be generated; for each dimension, if this random number is less than $pc(i)$(Eq 5), then another two particles' historical optimum values will be compared, with the better one chosen for the exemplar in the d^{th} dimension. If all the exemplars of a particle are its own *pbest*, then we will randomly choose one dimension to learn from another particle's *pbest*'s corresponding dimension.

$$pc(i) = 0.05 + 0.45 * \frac{\exp(10 * (i-1)/(N-1)) - 1}{\exp(10) - 1} \quad (5)$$

2.3 Lagrange interpolation

The theory of Lagrange interpolation is to use a polynomial to represent the relationship between a number of things. For example, when observing a physical quantity, if we gain some different values at different places, then a polynomial can be simulated by Lagrange interpolation method. The aim of this method is mainly used for data fitting in engineering experiments. It is a kind of curve smoothly fitting method.

Generally, if the fitness $y_0, y_1 \cdots y_n$ at n+1 points $x_0, x_1 \cdots x_n$ of function $y = f(x)$ are known, then a polynomial $y = P_n(x)$ is considered, who occupies the n+1 points and whose number is no less than n.

s.t.$P_n(x_k) = y_k$, $k = 0, 1, 2 \cdots n$

If we want to estimate a point ξ, $\xi \neq x_i$, $i = 0, 1, 2 \cdots n$, then the fitness of $P_n(\xi)$ can be the approximate value of $f(\xi)$ In this case, we obtain

$$f(x) = \sum_{i=1}^{n} y_i P_i(x) + error, \quad (i = 1, 2, \cdots n) \quad (6)$$

Where,$error = \frac{f^{(n+1)}(\xi)}{(n+1)!}\omega_{n+1}(x)$, ξ is a parameter related with x, and $\omega_{n+1}(x) = (x - x_0)(x - x_1)\cdots(x - x_n)$

3. The proposed methods

3.1 Local search with Lagrange interpolation (LSLI)

The main idea of CLPSO's search method is to make the particle experience all the local optima as much as possible. Hence, the Eq 3 of CLPSO cuts off the global optima part compared with the Eq 1 of PSO. Moreover, after learning from the other particle's historical optima, CLPSO sets a gap value m to digest this information. It is no doubt that these procedure will slow the particle swarm convergence, but will be in favor of multimodal function solution.

To accelerate the convergence, we decide to add a kind of local search into CLPSO. By far, there are some kinds of efficient technique, i.e. sub-gradient [26][27][28], perturbation, mutation or chaotic search with neighborhood [29][30][31][32][33][34][35]. The technique of sub-gradient can find the convergence direction easily. However, for the discontinuous and non-differentiable problems, the direction obtained will mislead the convergence. In addition, the step size is hard to decide. The technique of perturbation with neighborhood is not influenced by the form of the function. However, this method has no convergence direction, and usually needs many additional function evaluations (FEs). Hence, we adapt the local search technique of Lagrange interpolation to weaken these problems.

For the j^{th} dimension of the *gbest*, we select three points to generate the information and perform Lagrange interpolation. One point is the *gbest* itself, another point and the last point are the perturbations nearby the *gbest*. The perturbation value is denoted by *delta*, shown in Eq 7.

$$delta = rand * \eta * v(i,j) \tag{7}$$

$$\begin{aligned} x_0(j) &= gbest(j); \\ x_1(j) &= gbest(j) + delta; \\ x_2(j) &= gbest(j) - delta; \end{aligned} \tag{8}$$

Where, $v(i, j)$ is the particle's speed who has the best fitness for each iteration; η is a very small coefficient. In this research, we set $\eta = 0.5/N$, N is the particle swarm size. In the j^{th} dimension space, the three points can generate a parabola by the Lagrange interpolation method, and the minimum point is desired. Eqs 9 and 10 is the Lagrange interpolation computation.

$$f(x) = y_0\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1\frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2\frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \tag{9}$$

Where, $y_0 = fitness(x_0)$, $y_1 = fitness(x_1)$, $y_2 = fitness(x_2)$. Let $I = (x_0 - x_1)(x_1 - x_2)(x_2 - x_0)$, after calculating, we obtain a quadratic polynomial.

$$\begin{aligned} f(x) &= ax^2 + bx + c \\ a &= \frac{1}{I}[(x_2 - x_1)y_0 + (x_0 - x_2)y_1 + (x_1 - x_0)y_2] \\ b &= -\frac{1}{I}[(x_2^2 - x_1^2)y_0 + (x_0^2 - x_2^2)y_1 + (x_1^2 - x_0^2)y_2] \\ c &= \frac{1}{I}[x_1x_2y_0(x_2 - x_1) + x_0x_2y_1(x_0 - x_2) + x_0x_1y_2(x_1 - x_0)] \end{aligned} \tag{10}$$

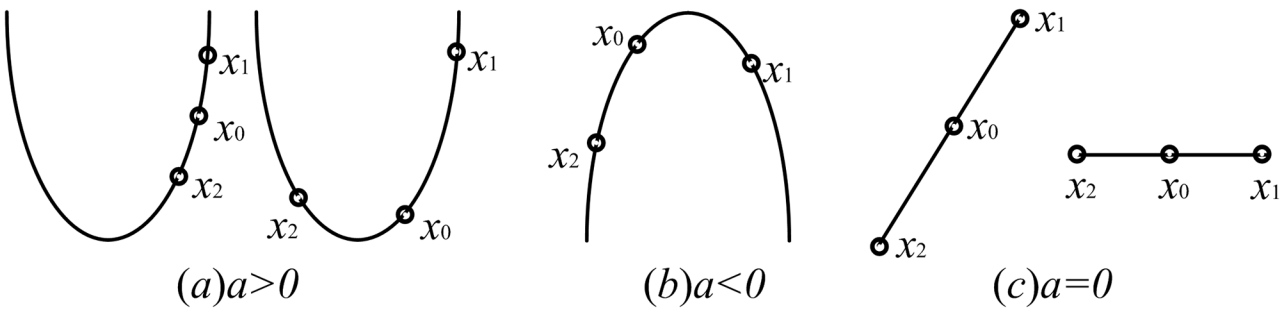

Fig 1. For$I \neq 0$, the different cases of the solution.

doi:10.1371/journal.pone.0154191.g001

Fig 1 shows the different cases of the desired solution. When $I \neq 0$, which means the three points are different, if $a > 0$, then two cases will happen: one is that the *gbest* is between $x1$ and $x2$, and another is that the *gbest* is on one side of $x1$ and $x2$. At this time, we will choose the minimum point -b/2a as the solution, then compare it with the *gbest*. If $a < 0$, then we will randomly generate a point near the smallest one by Eq 11. Where, $x_{\min}$, x_{mid} and $x_{\max}$ are the resort points of $x0$, $x1$ and $x2$ according to their fitness separately. If a = 0, then the three points form a line. If a = 0 and b = 0, this means $y0 = y1 = y2$, then we will randomly select a position from the area, denoted as $x3$ (Eq 12). Where, $\frac{(x_{\max}+x_{\min})}{2}$ is the search center, and because $-0.5 < rand - 0.5 < 0.5$, the random search radius is from 0 to half of the area. If a = 0 and b! = 0, then the solution will be chosen by Eq 11. When $I = 0$, which means $v(i, j) = 0$, then the procedure will be terminated.

$$x_3 = x_{\min} + rand * C_3 * (x_{\min} - x_{mid}) + rand * C_4 * (x_{\min} - x_{\max}) \tag{11}$$

$$x_3 = \frac{(x_{\max} + x_{\min})}{2} + (rand - 0.5)(x_{\max} - x_{\min}) \tag{12}$$

The flowchart of Lagrange interpolation is shown in Fig 2.

Comparing with other local search techniques, Lagrange interpolation has three characteristics. First, this method has a very fast convergence speed, especially for uni-modal functions. For example, for the Sphere function computation, whose global optima is $[0, 0]^{10}$, supposing that the *gbest* is $[1, 1]^{10}$, and the *delta* is 2. After one Lagrange interpolation computation, we obtain the next *gbest* is $[0, 0]^{10}$. Second, for each dimension Lagrange interpolation, it will cost three additional FEs, and for D dimension problem, the additional FEs of the *gbest* will be 3*D. Third, the behaviour of Lagrange interpolation is only a local search, it will not broken the diversity of the whole particle swarm, hence it will remain the CLPSO's search ability for multi-modal functions.

3.2 Lagrange interpolation learning (LIL)

In CLPSO, if the i^{th} particle's *pbest* does not update for a certain number of times, then another two particles' *pbest* will be chosen for comparison, and the better one will be the exemplar. However, if the exemplar is still worse than the i^{th} particle's *pbest*, then this particle will remain stagnant until the next $flag(i) > m$, with a large probability. Hence, the best manner to improve the search efficiency is to set the *gbest* as the exemplar. Nevertheless, the so-called *gbest* in the computation is not the real *gbest* rather, it can be a best one of many local optima that the

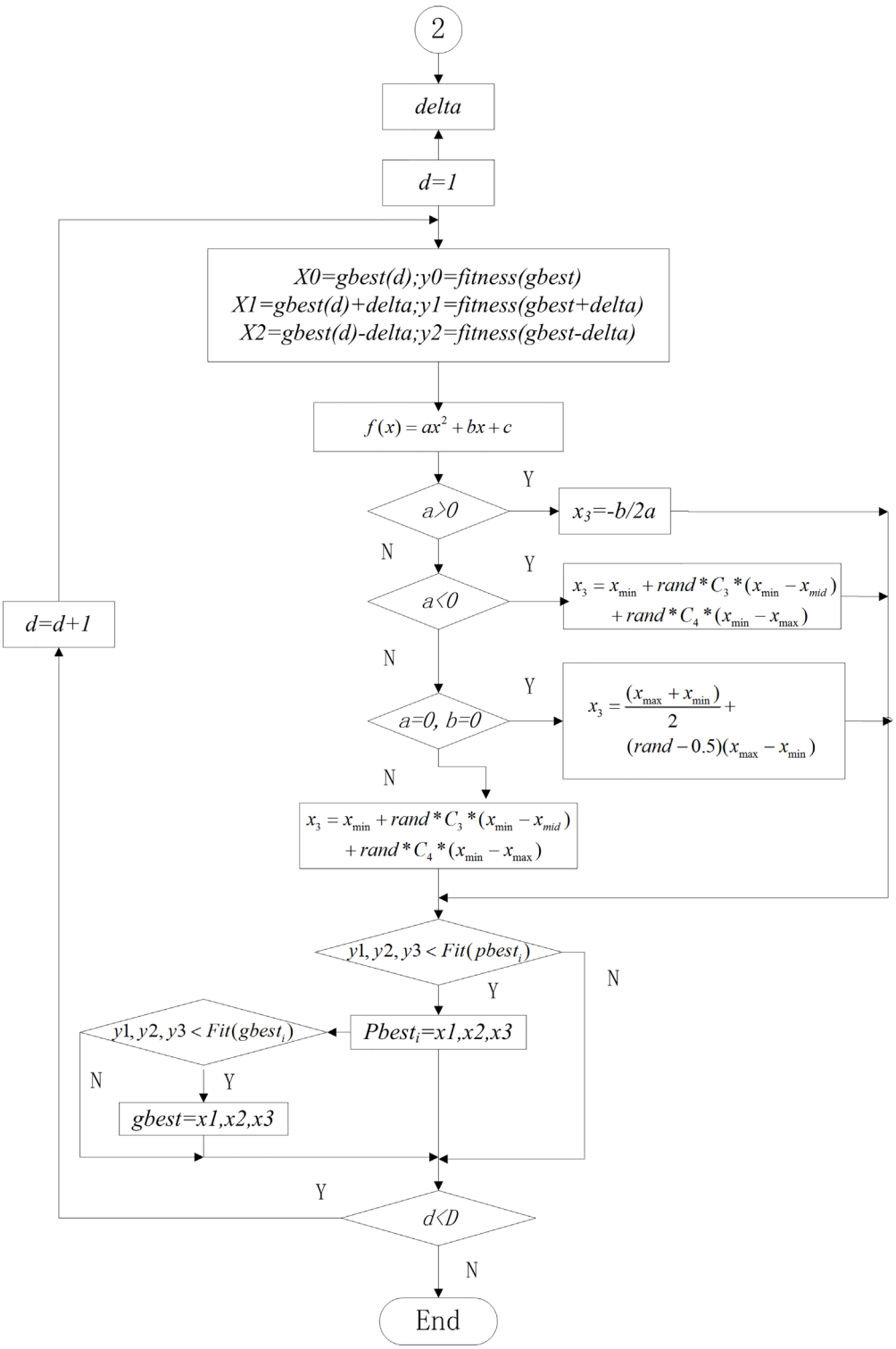

Fig 2. The flowchart of LSLI.

doi:10.1371/journal.pone.0154191.g002

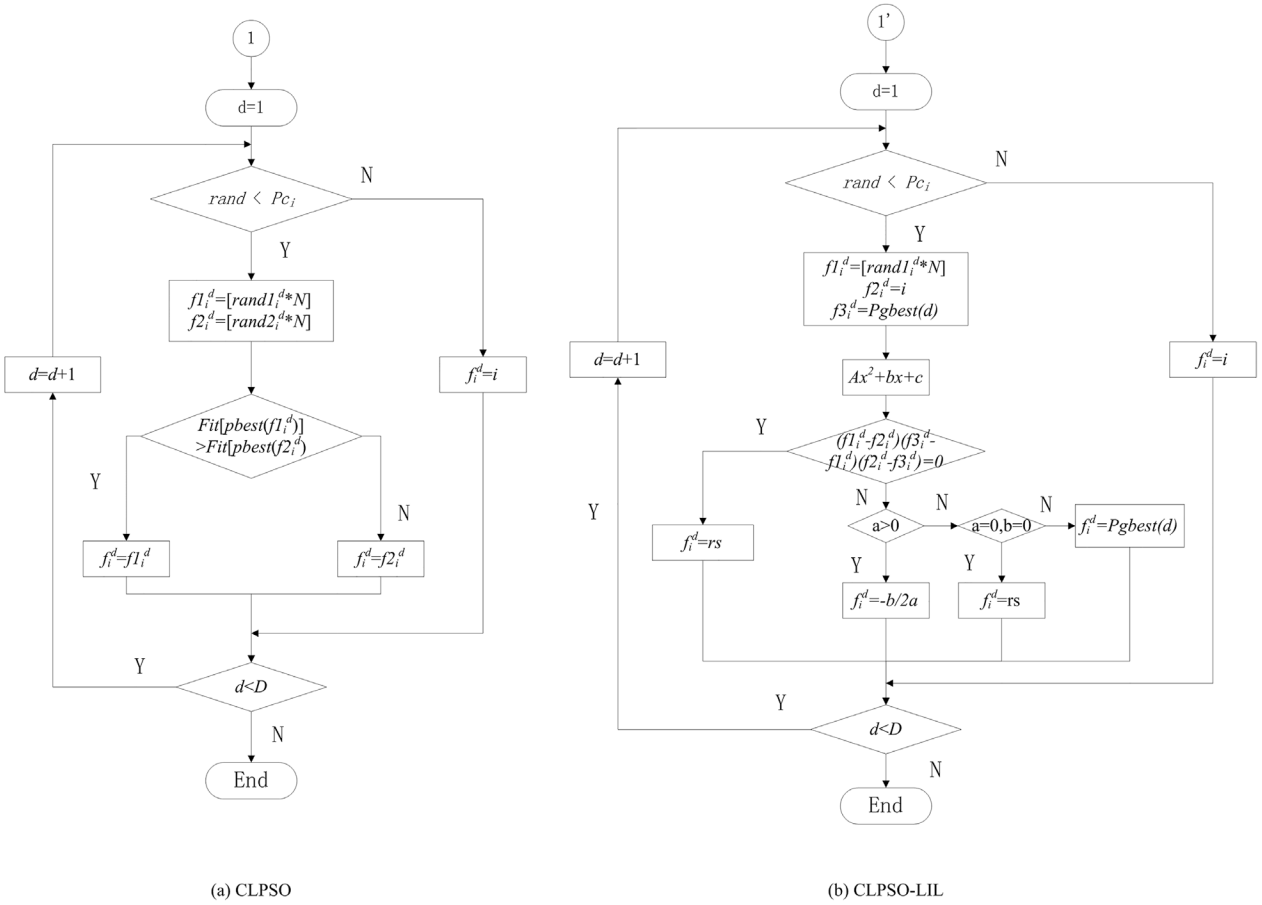

Fig 3. Selection of the exemplar dimensions for particle i. (a)CLPSO (b)CLPSO-LIL.

doi:10.1371/journal.pone.0154191.g003

program runs until now. If we set *gbest* as the exemplar directly, as in the traditional PSO, then a premature convergence will occur.

To avoid both the particle's stagnant nature and premature convergence, we ensure that the exemplar's information has two characteristics. First, the fitness of the exemplar is at least better than the i^{th} particle's *pbest*. Second, the information from the exemplar has a diversity that cannot lead all the particles to prematurely fly into a same area. Hence, we decide to select three points to generate the information mentioned above. One point is the i^{th} particle's *pbest* itself, another point is *gbest*, and the last point is the $rand^{th}$ particle's *pbest*, except for the i^{th} particle. In the d^{th} dimension space, the three points can generate a parabola by the Lagrange interpolation method, and the minimum point is desired. Fig 3 shows the difference between CLPSO and LIL.

This search behaviour has three strengths: First, the exemplar is the minimum from Lagrange interpolation with three points which are two *pbests* and one *gbest*, this process ensures the learning direction is always flying to a theoretical point which is better than *gbest*. Second, for most of the local search, additional time must be spent on computing the function evaluations (FEs) to obtain some key information, whereas LIL does not require any additional FEs. Third, after performing the LSLI mentioned in Section 3.1, we obtained a better *gbest*, then the LIL can share this information to other particles as soon as possible.

3.3 Parametric settings

To a convenient computation, we plan to run LSLI for N times, which is equal to the particle swarm size. However, To a fair comparison, noticing that there will be 3*D additional FEs for each LSLI, and we hope the total FEs of all compared algorithms are equal, thus we short the max iteration number (*max_gen*) to *max_gen* − 3 * *D*. The FEs cost in LSLI are 3*D*N, and FEs cost on other part are (*max_gen* − 3 * *D*)*N, plus them, we get the total FEs *max_gen* * *N*, which is the same with CLPSO.

When to run LSLI? we set a gap g, g = floor(maxDT/N), where, *maxDT* = *max_gen* − 3 * *D*. Hence, when the particle swarm updates for g times by Eqs 3 and 4, LSLI will run for one time.

In CLPSO, the learning probability pc(i) is set as Eq 5. In this research, we chose a linear probability form the Ref. [28] to increase the learning chance.

$$pc(i) = 0.05 + 0.45 * i/N \tag{13}$$

To compare with OLPSO [36], in this algorithm, we chose both c1 and c2 to be 2, w0 = 0.9 and w1 = 0.4. The setting of the bounds of the search space and the velocity of any particle affect the search procedure; hence we chose the same velocity boundary setting as that used in most of the algorithms.

$$V_{\max} = \alpha(x_{\max} - x_{\min}), \quad V_{\min} = -V_{\max}$$

Where, α = 0.2. The flow chart of LILPSO is shown in Fig 4.

4. Numerical experiments

Seventeen functions are collected from [17][19][37][38][39], as presented in Table 1, where, F1, F3, F4 and F5 are uni-modal functions; Rosenbrock (F2) is a multi-modal function that has a narrow valley and hard to achieve the global optimum; F5 is a noisy function who has a discrete problem; un-rotated multi-modal functions include F6 F11; F12 and F13 are rotated multi-modal functions; F14 F16 are shifted rotated multimodal functions. The orthogonal matrix M is generated according to [38].

Since some other algorithms, such as PSO-cf-local [36], UPSO [40], FIPS [41] and DMSPSO [42], are proven to be less superior to CLPSO in reference [19]; hence, in this research, we just need to compare LILCLPSO with CLPSO, ECLPSO, OLPSO and DNLPSO, whose iterative forms are presented in Table 2. The parametric setting of ECLPSO is the same as that of ECLPSO-4 in [22]. Because we do not know the exact orthogonal matrix for OLPSO, we introduce the result directly from the reference [19]. For DNLPSO, we don't know the exact topology data, hence we introduce the result directly from the reference [26]. To proof the performance of LSLI and LIL, we test two algorithms, called LILPSO1 and LILPSO2, where, LILPSO1 just adapt LSLI technique, and LILPSO2 adapt both LSLI and LIL technique.

For 10 D problems, the following parameters are used: particle number = 50, iteration number = 2000, and FEs = 100,000; for 30 D problems, the following parameters are used: particle number = 40, iteration number = 5000, and FEs = 200,000; for 50 D problems, the following parameters are used: particle number = 100, iteration number = 5000, and FEs = 500,000. For each function, each algorithm runs for 25 times, and the solutions are analysed using the two-tailed t-test, with the confidence level of 0.05, '+', '-' and '=' denote that LILPSO is better, worse and equal to other algorithms statistically, respectively.

Tables 3–8 list the results tested in 10D, 30D and 50D, respectively, and Fig 5(a)-5(f) show these algorithms' convergence curves for some different functions.

For the uni-modal and low dimension problems, DNLPSO has the better performance relatively. For the noisy function F5, LILPSO has the better performance. For uni-modal function

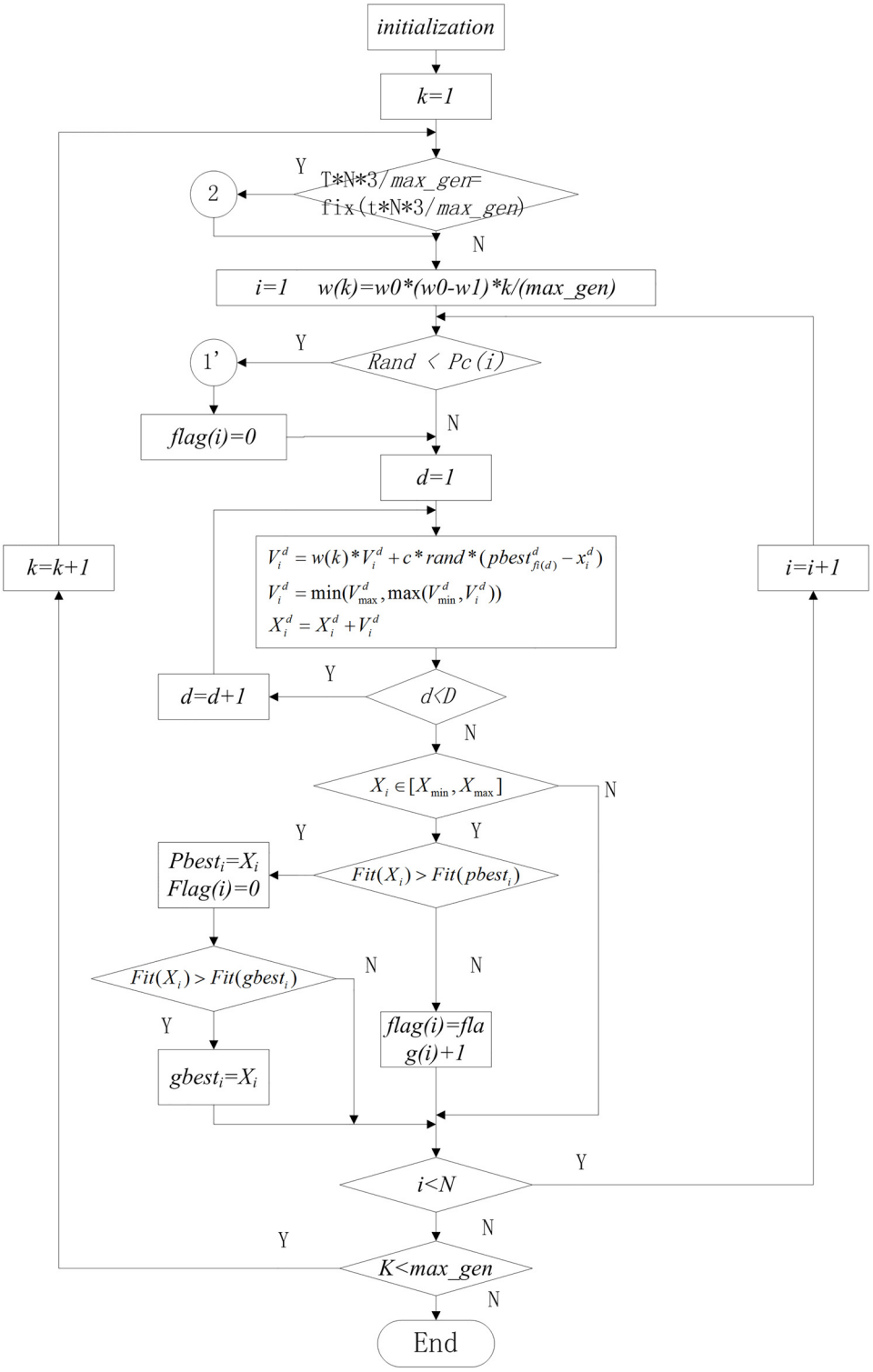

Fig 4. The flowchart of LILPSO.

doi:10.1371/journal.pone.0154191.g004

Table 1. Details of benchmarks.

NO.	Func.name	Expression	Box constraint	optimum
F1	Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$	[–100, 50]D	[0, 0]D
F2	Rosenbrock	$f(x)=\sum_{i=1}^{n-1}100(x_{i+1}-x_i^2)^2+(x_i-1)^2$	[–30, 30]D	[0, 0]D
F3	Step	$f(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	[–100, 100]D	[0, 0]D
F4	Schwefel's P2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[–10, 10]D	[0, 0]D
F5	Noise Quadric	$f(x)=\sum_{i=1}^{n}ix_i^4+random(0,1)$	[–1.28, 1.28]D	[0, 0]D
F6	A generalized penalized	$f(x)=\frac{\pi}{n}(10\sin^2(\pi y_1)+\sum_{i=1}^{n}(y_i-1)(1+10\sin^2(\pi y_{i+1}))+(y_n-1)^2)+\sum_{i=1}^{n}u(x_i,10),$ $y_i=1+(x_i+1)/4,\ \ u(x_i,a)=\begin{cases}100(x_i-a)^4, & if\ \ x_i>a\\ 0, & if\ \ -a\le x_i\le a\\ 100(-x_i-a)^4 & if\ \ x_i<-a\end{cases}$	[–50, 50]D	[0, 0]D
F7	Another generalized penalized	$f(x)=(\sin^2(3\pi x_1)+(x_n-1)^2(1+\sin^2(2\pi x_n)+\sum_{i=1}^{n-1}(x_i-1)^2(1+\sin 3\pi x_i))/10+\sum_{i=1}^{n}u(x_i,5)$	[–50, 50]D	[0, 0]D
F8	Ackley	$f(x)=20+e-20*\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	[–32, 32]D	[0, 0]D
F9	Rastrigin	$f(x)=\sum_{i=1}^{n}x_i^2-10\cos 2\pi x_i+10$	[–5, 5]D	[0, 0]D
F10	Griewank	$f(x)=1+\frac{\sum_{i=1}^{n}(x_i-100)^2}{4000}-\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)$	[–600, 200]D	[0, 0]D
F11	Schwefel	$f(x)=418.9829-\sum_{i=1}^{n}x_i\sin(\sqrt{\lvert x_i\rvert})$	[–500, 500]D	[420, 96]D
F12	Ackley-Rotated	$f_{12}(y)=f_8(y),\ y=Mx$	[–32, 32]D	[0, 0]D
F13	Rastrigin-Rotated	$f_{13}(y)=f_9(y),\ y=Mx$	[–5, 5]D	[0, 0]D
F14	Griewank-Rotated	$f_{14}(y)=f_{10}(y),\ y=Mx$	[–600, 200]D	[0, 0]D
F15	Ackley-Rotated-shifted	$f_{15}(y)=f_8(y)+f\_bias,\ \ y=M(\vec{x}-\vec{o})$	[–32, 32]D	$\vec{o}$
F16	Rastrigin-Rotated-shifted	$f_{16}(y)=f_8(y)+f\_bias,\ \ y=M(\vec{x}-\vec{o})$	[–5, 5]D	$\vec{o}$
F17	Griewank-Rotated-shifted	$f_{17}(y)=f_{10}(y)+f\_bias,\ \ y=M(\vec{x}-\vec{o})$	[–600, 200]D	$\vec{o}$

doi:10.1371/journal.pone.0154191.t001

Table 2. Iterative forms of each algorithms.

Algorithm	Iterative forms
CLPSO [10]	$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1(p_{id}(t) - x_{id}(t))$
ECLPSO [33]	$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1\left(p_{id}(t) + \eta\left(\frac{p_{d\_low}+p_{d\_up}}{2} - p_{id}(t)\right) - x_{id}(t)\right)$
OLPSO [38]	$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1(p_{id}(t) - x_{id}(t))$
DNLPSO [31]	$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1(p_{id}(t) - x_{id}(t)) + c_2 r_2(p_{gd}(t) - x_{id}(t))$
LILPSO	$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1(p_{id}(t) - x_{id}(t))$

doi:10.1371/journal.pone.0154191.t002

Table 3. Results for D = 10, N = 50, FEs = 100,000.

Function		Algorithm CLPSO	ECLPSO	DNLPSO	LILPSO1	LILPSO2
F1	mean	4.94E−19	1.78E−30	**5.71E−177**	1.64E−58	1.11E−73
	sd	2.19E−37	2.34E−60	1.11E−123	7.11E−116	**6.27E−146**
	ttest	+	+	−	+	
F2	mean	1.26E+00	1	**1.86E−03**	1.00E+00	3.28E−01
	sd	1.79E+00	3.80E−09	2.64E−02	**6.32E−10**	1.71E−01
	ttest	+	+	−	+	
F3	mean	2.05E−18	1.03E−30	**0.00E+00**	**0.00E+00**	**0.00E+00**
	sd	2.27E−36	9.88E−61	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	=	
F4	mean	1.50E−11	6.22E+00	**1.96E−66**	2.29E−13	1.50E−16
	sd	3.73E−23	5.67E+01	**3.31E−54**	1.54E−27	1.61E−33
	ttest	+	+	−	+	
F5	mean	5.00E−03	2.30E−03	5.21E−01	4.50E−03	**1.40E−03**
	sd	3.34E−06	1.74E−06	1.89E−01	3.40E−06	**1.99E−07**
	ttest	+	+	+	+	
F6	mean	2.20E−18	2.85E−29	**4.71E−32**	**4.71E−32**	**4.71E−32**
	sd	4.46E−36	7.99E−57	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	=	
F7	mean	2.02E−16	1.41E−27	**1.35E−32**	2.45E−32	**1.35E−32**
	sd	3.24E−32	5.91E−54	**0.00E+00**	1.44E−65	**0.00E+00**
	ttest	+	+	=	=	
F8	mean	1.94E−08	3.55E−15	**3.26E−15**	2.58E−11	1.84E−14
	sd	1.49E−16	4.21E−30	1.83E−15	3.02E−22	**6.84E−28**
	ttest	+	+	=	+	
F15	mean	3.76E−02	2.31E+00	2.02E+01	6.27E−05	**2.01E−08**
	sd	4.01E−04	2.83E+01	4.04E−01	4.38E−09	**2.26E−16**
	ttest	+	+	+	+	
F16	mean	6.11E+00	7.09E+01	7.57E+00	2.71E+00	**2.67E+00**
	sd	7.02E+00	1.03E+02	**4.70E+00**	8.83E+00	6.65E+00
	ttest	+	+	+	+	
F17	mean	1.40E−01	1.84E+01	**8.07E−03**	1.52E−04	9.29E−02
	sd	3.10E−03	2.53E+02	1.14E−01	**2.01E−08**	1.09E−02
	ttest	+	+	=	−	

doi:10.1371/journal.pone.0154191.t003

Table 4. Results for D = 10, N = 50, FEs = 100,000.

Function		Algorithm CLPSO	ECLPSO	LILPSO1	LILPSO2
F9	mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	sd	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ttest	=	=	=	
F10	mean	**7.92E–05**	3.16E–01	1.26E–04	5.00E–03
	sd	**4.09E–08**	1.33E–02	6.90E–08	5.15E–05
	ttest	–	+	–	
F11	mean	**1.27E–04**	1.78E+03	**1.27E–04**	**1.27E–04**
	sd	6.34E–26	3.26E+04	**0.00E+00**	**0.00E+00**
	ttest	=	+	=	
F12	mean	3.08E–02	3.66E+00	2.46E–06	**2.87E–08**
	sd	1.38E–04	4.42E+01	2.33E–12	**6.09E–16**
	ttest	+	+	+	
F13	mean	6.41E+00	6.15E+01	**1.35E+00**	6.95E+00
	sd	4.49E+00	8.91E+01	**2.16E+00**	5.45E+01
	ttest	–	+	–	
F14	mean	1.64E–01	1.06E+01	**6.27E–05**	5.34E–02
	sd	6.20E–03	1.08E+02	**4.38E–09**	5.80E–03
	ttest	+	+	+	

doi:10.1371/journal.pone.0154191.t004

F3, DNLPSO and LILPSO have the equal solutions. Because DNLPSO's iterative form contains the gbest part, it will have a fast convergence for uni-modal functions without a doubt. Besides, the neighbourhoods topology behaviour plays a role of decreasing the search space actually. However, for the high dimension problems, i.e. 50D, LILPSO has all the best solutions, which illustrates the Lagrange interpolation technique has a fast convergence performance for complex high dimension problems.

For the multi-modal problems, DNLPSO is less superior to LILPSO for almost all the functions. Comparing OLPSO with LILPSO in 30D problems, for F2 and F8, LILPSO has the better solutions; for F6, F7, F9, F10 and F11, LILPSO has the equal solutions with OLPSO statistically, which illustrate that the Lagrange interpolation technique can help accelerating the pbest's convergence when performing a local search. For F12, F13 and F14, OLPSO is superior to LILPSO, which illustrates that LSLI is restricted to rotated problems, although it is still better than CLPSO.

Comparing LILPSO1 with LILPSO2, for the most problems, LILPSO2 is superior to LILPSO1, which illustrates that LIL can help sharing the LSLI's information to accelerate the convergence. Meanwhile, unlike the gbest's part of DNLPSO, LIL neither break the diversity of the particle, nor lead the particle to premature. However, for the rotated functions such as F13, F14, F16 and F17, the solutions of LILPSO1 are better than LILPSO2, which illustrates that LIL is not suitable for solving the rotated problems either.

Comparing CLPSO with LILPSO, LILPSO has all the better solutions than CLPSO, which illustrates that the Lagrange interpolation is a stable and efficient local search technique.

Comparing OLPSO with LILPSO, they both inherit the advantage of CLPSO in solving the multi-modal problems, nevertheless, LILPSO is obviously superior to OLPSO in solving the uni-modal problems. Moreover, OLPSO-L needs $O(2^{\log_2 (D+1)} D + ND)$ memory space to store its algorithm related data structures, which means longer cost time for complex real

Table 5. Results for D = 30, N = 40, FEs = 200,000.

Function		Algorithm CLPSO	ECLPSO	OLPSO-G	OLPSO-L	LILPSO1	LILPSO2
F1	mean	5.66E−15	1.32E−26	4.12E−54	1.11E−38	3.00E−66	**4.19E−87**
	sd	1.33E−29	1.63E−51	6.34E−54	1.28E−38	4.31E−131	**8.79E−173**
	ttest	+	+	+	+	+	
F2	mean	8.28E+00	**1.00E+00**	2.15E+01	1.26E+00	**1.00E+00**	**1.00E+00**
	sd	3.57E+01	**0.00E+00**	2.99E+01	1.40E+00	2.39E−13	1.90E−05
	ttest	+	=	+	+	=	
F4	mean	3.72E−09	7.41E+01	**9.85E−30**	7.67E−22	4.81E−13	4.86E−17
	sd	2.12E−18	1.33E+02	1.01E−29	5.63E−22	8.61E−27	**1.83E−33**
	ttest	+	+	−	−	+	
F5	mean	1.57E−02	**6.40E−03**	1.16E−02	1.64E−02	1.05E−02	8.10E−03
	sd	1.31E−05	5.48E−06	4.10E−03	3.25E−03	1.65E−05	**4.53E−06**
	ttest	+	=	+	+	+	
F6	mean	1.63E−15	6.81E−22	1.59E−32	**1.57E−32**	**1.57E−32**	**1.57E−32**
	sd	1.05E−30	6.94E−42	1.03E−33	2.79E−48	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	=	=	
F7	mean	2.05E−12	5.83E−20	4.39E−04	1.57E−32	1.33E−31	**1.35E−32**
	sd	3.59E−24	5.47E−38	2.20E−03	2.79E−48	1.90E−62	**0.00E+00**
	ttest	+	+	+	=	+	
F8	mean	6.92E−07	2.20E−05	7.98E−15	4.14E−15	2.25E−11	**2.84E−15**
	sd	1.21E−13	9.75E−09	2.03E−15	**0.00E+00**	5.35E−22	2.52E−30
	ttest	+	+	+	+	+	
F9	mean	1.77E−15	1.91E+00	2.17E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**
	sd	0.00E+00	6.95E+01	1.07E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ttest	+	+	+	=	=	
F10	mean	3.35E−10	1.88E+02	4.83E−03	**0.00E+00**	**0.00E+00**	**0.00E+00**
	sd	4.02E−19	1.15E+03	8.63E−03	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ttest	+	+	+	=	=	
F11	mean	3.81E−04	6.58E+03	3.84E+02	**3.81E−04**	2.36E+01	**3.81E−04**
	sd	3.41E−23	2.53E+05	2.17E+02	**0.00E+00**	2.80E+03	**0.00E+00**
	ttest	=	+	+	=	+	
F12	mean	3.36E+00	1.83E+01	7.69E−15	**4.28E−15**	6.40E−03	2.60E−03
	sd	5.73E−01	1.59E−01	1.78E−15	**7.11E−16**	6.49E−15	9.19E−06
	ttest	+	+	−	−	+	
F13	mean	3.92E+01	3.15E+02	**4.60E+00**	5.34E+01	2.92E+01	3.56E+01
	sd	5.80E+01	4.78E+02	**1.28E+01**	1.33E+01	9.63E+01	6.96E+02
	ttest	+	+	−	+	−	
F14	mean	1.06E+00	2.09E+02	1.68E−03	**4.19E−08**	7.69E−04	7.06E−02
	sd	2.10E−03	8.32E+01	4.13E−03	**2.06E−07**	8.66E−07	7.70E−03
	ttest	+	+	−	−	−	

doi:10.1371/journal.pone.0154191.t005

Table 6. Results for D = 30, N = 40, FEs = 200,000.

Function		Algorithm CLPSO	ECLPSO	LILPSO1	LILPSO2
F3	mean	1.41E−14	1.13E−19	**0.00E+00**	**0.00E+00**
	sd	4.76E−29	2.53E−37	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	
F15	mean	3.24E+00	1.87E+01	3.60E−03	**1.10E−03**
	sd	3.78E−01	2.50E−01	2.81E−05	**2.60E−06**
	ttest	+	+	+	
F16	mean	3.54E+01	3.53E+02	**2.36E+01**	4.66E+01
	sd	2.25E+01	8.27E+02	**2.19E+01**	7.91E+01
	ttest	−	+	=	
F17	mean	1.06E+00	2.15E+02	**1.40E−02**	3.46E−02
	sd	4.70E−03	8.67E+01	**7.11E−04**	4.20E−03
	ttest	+	+	−	

doi:10.1371/journal.pone.0154191.t006

Table 7. results for D = 50, N = 100, FEs = 500,000.

Function		Algorithm CLPSO	ECLPSO	DNLPSO	LILPSO1	LILPSO2
F1	mean	1.07E−08	8.06E+03	9.44E−74	2.60E−57	**4.77E−74**
	sd	5.82E−18	1.26E+07	9.13E−32	1.22E−113	**5.61E−147**
	ttest	+	+	+	+	
F2	mean	3.16E+01	1.08E+01	**1.86E−03**	1.00E+00	2.28E−01
	sd	2.04E+02	8.00E+02	2.64E−02	**6.23E−14**	1.89E−01
	ttest	+	+	−	+	
F3	mean	3.19E−08	2.03E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**
	sd	4.15E−17	1.14E+08	**0.00E+00**	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	=	
F4	mean	2.59E−05	1.27E+02	7.02E−09	8.85E−09	**3.84E−13**
	sd	2.80E−11	2.38E+02	2.29E−06	1.99E−18	**5.09E−25**
	ttest	+	+	+	+	
F5	mean	2.69E−02	1.97E−02	6.11E−01	1.32E−02	**9.30E−03**
	sd	1.64E−05	2.05E−05	2.20E−01	**1.92E−06**	8.08E−06
	ttest	+	+	+	+	
F6	mean	1.73E−09	1.35E+02	1.50E−32	1.28E−32	**9.42E−33**
	sd	3.25E−19	3.25E+05	0.00E+00	2.57E−66	**0.00E+00**
	ttest	+	+	+	+	
F7	mean	1.69E−06	1.92E+02	1.34E−32	8.75E−31	**9.42E−33**
	sd	2.23E−13	7.26E+05	0.00E+00	1.77E−60	**0.00E+00**
	ttest	+	+	+	+	
F8	mean	3.95E−04	1.58E+01	1.90E+01	8.13E−08	**3.05E−14**
	sd	1.20E−08	1.43E+01	0.00E+00	1.87E−15	**1.55E−27**
	ttest	+	+	+	+	
F15	mean	6.80E+00	1.81E+01	2.07E+01	**1.86E−01**	3.07E−01
	sd	2.47E−01	9.77E−02	1.13E−02	**2.86E−02**	3.61E−01
	ttest	+	+	+	−	
F16	mean	1.07E+02	1.65E+02	**3.40E+01**	8.83E+01	8.13E+01
	sd	5.02E+01	2.03E+02	**8.03E+00**	2.27E+02	1.22E+04
	ttest	+	+	−	=	
F17	mean	2.63E+00	4.40E+02	**1.11E−02**	2.75E−01	2.54E−01
	sd	1.61E−01	4.54E+01	**1.12E−02**	2.46E−02	4.04E−02
	ttest	+	+	−	=	

doi:10.1371/journal.pone.0154191.t007

Table 8. results for D = 50, N = 100, FEs = 500,000.

Function		Algorithm CLPSO	ECLPSO	LILPSO1	LILPSO2
F9	mean	1.22E−10	6.80E+01	**0.00E+00**	**0.00E+00**
	sd	7.80E−22	4.54E+02	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	
F10	mean	2.56E−07	5.71E+02	**0.00E+00**	**0.00E+00**
	sd	1.34E−14	3.73E+04	**0.00E+00**	**0.00E+00**
	ttest	+	+	=	
F11	mean	6.59E−04	1.44E+04	**6.36E−04**	**6.36E−04**
	sd	1.11E−10	2.77E+06	2.53E−17	**0.00E+00**
	ttest	+	+	=	
F12	mean	8.12E+00	1.74E+01	**7.88E−02**	4.22E+00
	sd	9.58E−01	4.08E−01	**1.50E−03**	7.79E+01
	ttest	+	+	−	
F13	mean	1.09E+02	4.89E+02	**9.51E+01**	1.12E+02
	sd	1.07E+02	9.71E+02	8.31E+01	**2.64E+03**
	ttest	−	=	−	
F14	mean	2.54E+00	4.33E+02	**2.28E−01**	2.88E−01
	sd	9.47E−02	8.61E+01	**3.30E−03**	4.00E−02
	ttest	+	+	−	

doi:10.1371/journal.pone.0154191.t008

world problems. In contrast, LILPSO just needs a small number of memory space to store some related variables, i.e. x0, y0, x1, y1, x2, y2, a, b and c.

5. Application for PID control

The fan speed system controlled by oil in air turbofan launch is taken for example. The transfer function model of the system is:

$$\frac{1.192s + 6.273}{s^2 + 7.167s + 12.84}$$

PID discretion control equation is

$$Kp * e(k) + Ki * \sum_{i=0}^{k} e(i) + Kd[e(k) - e(k-1)]$$

The objective function is:

$$T = \int_0^\infty (\omega 1 \cdot |e(t)| + \omega 2 \cdot u^2(t))dt + \omega 3 \cdot t_u$$

Where, $|e(t)|$ is the error, t_u is the rise time, $u(t)$ is the output of the controller, $\omega 1$, $\omega 2$, $\omega 3$ are the weight. To solve the problem of system overshoot, use the punish function, once the system overshoot happens, the objective function will be:

$$T = \int_0^\infty (\omega 1|e(t)| + \omega 2u^2(t) + \omega 4|ey(t)|)dt + \omega 3 \cdot t_u$$

Where, $\omega 4 >> \omega 1$, $ey(t) = y(t) - y(t-1)$, $y(t)$ is the output of the object controlled. In this example, $\omega 1 = 0.5$, $\omega 2 = 1$, $\omega 3 = 1$, $\omega 4 = 200$, the range of Kp, Ki, Kd is respectively [0.2, 10], [1,

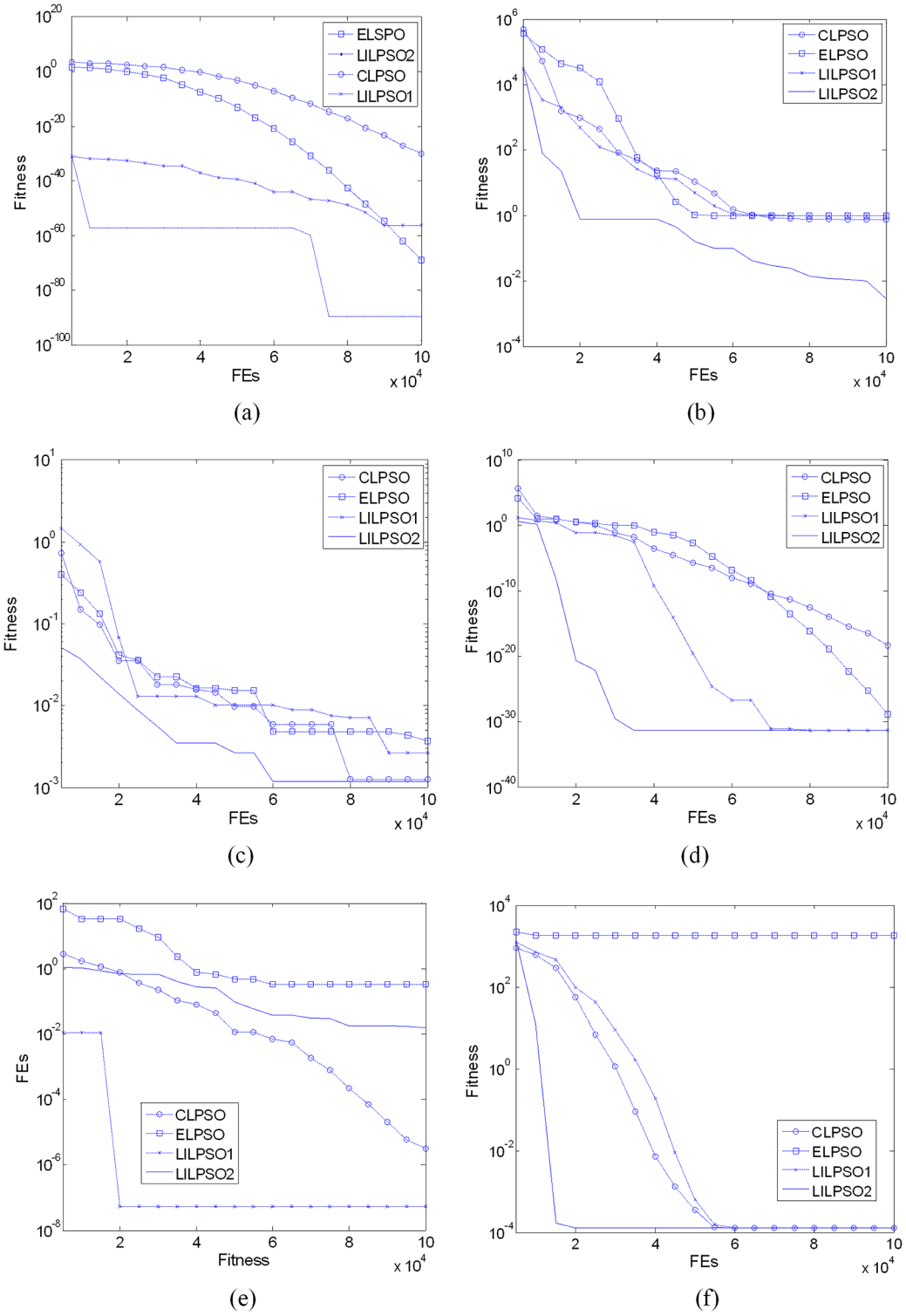

Fig 5. The comparison on convergence. (a) Sphere (b) Rosenbrock (c) Noise Quadric (d) Penalized (e) Griewank (f) Schwefel.

doi:10.1371/journal.pone.0154191.g005

Table 9. PID Results optimized by some algorithms.

Variables	CLPSO [10]	ECLPSO [33]	LILPSO2
Kp	3.0934	2.719	4.402
Ki	17.2368	16.817	24.58
Kd	0.042	6.13E–05	0.0964
overshoot	10.79%	11.75%	9.90%
Mean error	0.0099	0.0101	0.0079
T	2261	2267	2218

doi:10.1371/journal.pone.0154191.t009

50], [1e-7, 1e-1], and the overshoot should be smaller than 20%. Hence, the problem is:

$$\begin{aligned} \min \quad & T \\ s.t. \quad & 0.2 \leq Kp \leq 10 \\ & 1 \leq Ki \leq 50 \\ & 1e-7 \leq Kd \geq 1e-1 \\ & overshoot \leq 0.2 \end{aligned} \tag{14}$$

The parameters are set as: swarm size N = 30, function evaluations FEs = 3000. The results are shown in Table 9. It illustrates that the results optimized by LILPSO2 have the smallest goal function value, and the mean error. Hence, LILPSO2 algorithm is more efficient.

6. Conclusions

In this study, we proposed a novel method known as LILPSO to improve upon the state-of-the-art CLPSO method. First, the Lagrange interpolation approach is introduced to perform a local search near the gbest, and help accelerating convergence. Second, this technique is introduced to replace the simple comparison used in CLPSO, to achieve a better exemplar. After performing numerical experiments, LILPSO was proven to be superior to CLPSO, ECLPSO, DNLPSO, OLPSO for most of the test functions considered. The Lagrange interpolation was proven to be an efficient local search approach except for rotated problems. The future work is to use this method into other fields.

Supporting Information

S1 Table. The comparison results for CLPSO, ECLPSO, DNLPSO, LILPSO1 and LILPSO2.
(XLS)

Author Contributions

Conceived and designed the experiments: ZK SJ. Performed the experiments: ZK NK. Analyzed the data: ZK LS. Contributed reagents/materials/analysis tools: ZK NK LS. Wrote the paper: LS.

References

1. Kennedy J., Eberhart R.C. Particle swarm optimization. in Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, 1995; pp. 1942–1948.

2. Kirkpatrick SC, Gelatt CD, Vecchi MP. Optimization by Simulated Annealing. Science. 1983; 220 (4598):671–80. doi: 10.1126/science.220.4598.671 PMID: 17813860
3. Haddow BPC, Tufte G, editors. Goldberg D.E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. In Proceedings of the 2000 Congress on Evolutionary Computation CEC00; 2010.
4. Shi Y, Eberhart R C. A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation. Proceedings of the Piscataway, 1998; 1998b pp. 6973.
5. By Y, Eberhart R, editors. 1998). " Parameter selection in particle swarm optimization. Proceedings of the 7th Annual Conference on Evolutionary Programming; 2010.
6. HUANG gang, LONG yuanming, LI jinhang, et al. Adaptive Particle Swarm Optimization. IEEE Transactions on Systems Man and Cybernetics Part B. 2009; 39(6):1362–81. doi: 10.1109/TSMCB.2009.2015956
7. Ardizzon G, Cavazzini G, Pavesi G. Adaptive acceleration coefficients for a new search diversification strategy in particle swarm optimization algorithms. Information Sciences. 2015; 299(C):337–78. doi: 10.1016/j.ins.2014.12.024
8. Arpso T. A Diversity-Guided Particle Swarm Optimizer—the ARPSO. Deptcomputsciunivof Aarhus. 2002.
9. Shieh HL, Kuo CC, Chiang CM. Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. Applied Mathematics and Computation. 2011; 218(8):4365–83. doi: 10.1016/j.amc.2011.10.012
10. Leu MS, Yeh MF. Grey particle swarm optimization. Applied Soft Computing. 2012; 12(9):2985–96. doi: 10.1016/j.asoc.2012.04.030
11. Liu B, Wang L, Jin YH, Tang F, Huang DX. Improved particle swarm optimization combined with chaos. Chaos Solitons and Fractals. 2005; 25(5):1261–71. doi: 10.1016/j.chaos.2004.11.095
12. Xu W, Geng Z, Zhu Q, Gu X. A piecewise linear chaotic map and sequential quadratic programming based robust hybrid particle swarm optimization. Information Sciences. 2013; 218(1):85–102. doi: 10.1016/j.ins.2012.06.003
13. Noel MM. A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. Applied Soft Computing. 2012; 12(1):353–9. doi: 10.1016/j.asoc.2011.08.037
14. Gao Y, Du W, Yan G. Selectively-informed particle swarm optimization. Scientific Reports. 2015; 5.
15. Chen Liu, Wen-Bo Du, Wen-Xu Wang. Particle Swarm Optimization with Scale-Free Interactions. PLoS One 2014; 9, e97822. doi: 10.1371/journal.pone.0097822
16. Du WB, Gao Y, Liu C, Zheng Z, Wang Z. Adequate is better: particle swarm optimization with limited-information. Applied Mathematics and Computation. 2015; 268:832–8. doi: 10.1016/j.amc.2015.06.062
17. Liang JJ, Qin AK, Baskar S. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. Evolutionary Computation IEEE Transactions on. 2006; 10(3):281–95. doi: 10.1109/TEVC.2005.857610
18. Liang JJ, Suganthan PN, editors. Adaptive Comprehensive Learning Particle Swarm Optimizer with History Learning. Simulated Evolution and Learning, International Conference, Seal 2006, Hefei, China, October 15–18, 2006, Proceedings; 2006.
19. Zhan Z.H., Zhang J., Li Y., Shi Y.H. Orthogonal learning particle swarm optimization. IEEE Trans. Evol. Comput. 2011; 15: 832–847. doi: 10.1109/TEVC.2010.2052054
20. Zheng YJ, Ling HF, Guan Q. Adaptive Parameters for a Modified Comprehensive Learning Particle Swarm Optimizer. Mathematical Problems in Engineering. 2012; 68(3):939–55.
21. Nasir M, Das S, Maity D, Sengupta S, Halder U, Suganthan PN. A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. Information Sciences. 2012; 209 (5):16–36. doi: 10.1016/j.ins.2012.04.028
22. Yu X, Zhang X. Enhanced comprehensive learning particle swarm optimization. Applied Mathematics and Computation. 2014; 242:265–76. doi: 10.1016/j.amc.2014.05.044
23. Huang VL, Suganthan PN, Liang JJ. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. International Journal of Intelligent Systems. 2006; 21(2):209–26. doi: 10.1002/int.20128
24. Victoire TAA, Suganthan PN, editors. Improved MOCLPSO algorithm for environmental/economic dispatch. IEEE Congress on Evolutionary Computation, Cec 2007, 25–28 September 2007, Singapore; 2007.
25. Ali H, Khan FA. Attributed multi-objective comprehensive learning particle swarm optimization for optimal security of networks. Applied Soft Computing. 2013; 13(9):3903–21. doi: 10.1016/j.asoc.2013.04.015

26. Plevris V, Papadrakakis M. A Hybrid Particle Swarm and mdash; Gradient Algorithm for Global Structural Optimization. Computer-aided Civil and Infrastructure Engineering. 2011; 26:48–68.
27. Boyd S.. Sub-Gradient Methods. Stanford University, Stanford, CA. 2010.
28. Hu M, Wu T, Weir JD. An intelligent augmentation of particle swarm optimization with multiple adaptive methods. Information Sciences. 2012; 213(23):68–83. doi: 10.1016/j.ins.2012.05.020
29. Zhao X. Simulated annealing algorithm with adaptive neighborhood. Applied Soft Computing. 2011; 11 (2):1827–36. doi: 10.1016/j.asoc.2010.05.029
30. Michalewicz Z.. Genetic Algorithms + Data Structures = Evolution Programs, third ed. Springer-Verlag, London, UK. 1996.
31. Ni J.C., Li L., Qiao F.I., Wu Q.D. A novel memetic algorithm based on the comprehensive learning PSO. in Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, Brisbane, Australia. 2012; pp. 1–8.
32. Kennedy J, Mendes R, editors. Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. Soft Computing in Industrial Applications, 2003 SMCia/03 Proceedings of the 2003 IEEE International Workshop on; 2003.
33. Kennedy J, editor Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. Evolutionary Computation, 1999 CEC 99 Proceedings of the 1999 Congress on; 1999.
34. Yildiz AR, Saitou K. Topology Synthesis of Multicomponent Structural Assemblies in Continuum Domains. Journal of Mechanical Design. 2011; 133(1):788–96. doi: 10.1115/1.4003038
35. Wang H, Sun H, Li C, Rahnamayan S, Pan JS. Diversity enhanced particle swarm optimization with neighborhood search. Information Sciences An International Journal. 2013; 223(2):119–35. doi: 10.1016/j.ins.2012.10.012
36. Kennedy J. and Mendes R. "Population structure and particle swarm performance", in Proc. IEEE Congr. Evol. Comput. Honolulu, HI. 2002; pp. 1671–1676.
37. Nobuhiro I, Keiichiro Y, Genki U. Dynamic parameter tuning of particle swarm optimization. Ieej Transactions on Electrical and Electronic Engineering. 2006; 1(1):353–63.
38. Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. Biosystems. 1996; 39(3):263–78. doi: 10.1016/0303-2647(96)01621-8 PMID: 8894127
39. Wang Y, Li B, Weise T, Wang J, Yuan B, Tian Q. Self-adaptive learning based particle swarm optimization. Information Sciences. 2011; 181(20):4515–38. doi: 10.1016/j.ins.2010.07.013
40. Parsopoulos KE, Vrahatis MN. 'Unified particle swarm optimization in dynamic environments. Eds): Lecture Notes in Computer Science (LNCS. 2005; 4(2):590–9.
41. Wolpert DH, Macready WG. Macready W.G.: No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation. 1997; 1:67–82. doi: 10.1109/4235.585893
42. Liang JJ, Suganthan PN, editors. Dynamic multi-swarm particle swarm optimizer with local search. Evolutionary Computation, 2005 The 2005 IEEE Congress on; 2005.