A Particle Swarm Optimizer with Lifespan for Global Optimization on Multimodal Functions

Jun Zhang, *MIEEE* and Ying Lin, *Student Member, IEEE*

***Abstract*—The particle swarm optimizer (PSO) is a popular computing technique of swarm intelligence, known for its fast convergence speed and easy implementation. All the particles in the traditional PSO must learn from the best-so-far solution, which makes the best solution the leader of the swarm. This paper proposes a variation of the traditional PSO, named the PSO with lifespan (LS-PSO), in which the lifespan of the leader is adjusted according to its power of leading the swarm towards better solutions. When the lifespan is exhausted, a new solution is produced and it will conditionally replace the original leader depending on its leading power. Experiments on six benchmark multimodal functions show that the proposed algorithm can significantly improve the performance of the traditional PSO.**

I. Introduction

FUNCTION optimization is the model of many optimization problems in reality. Consider an optimization problem consisting of a mapping

$$f : X \to y,\ X \in \Psi, y \in \Phi, \tag{1}$$

where X is a solution in the numerical solution space Ψ and y is a objective value in the numerical range Φ. The goal of this problem is to find the best solution in Ψ. Thus, it can be transformed into a search for the maximum/minimum of the function *f*. Most real problems have many near-optimum solutions, which endows the corresponding f with multimodal distribution. Multimodal functions are more difficult to optimize than the unimodal ones, because the local optima usually trap the algorithms into prematurity. As many real world optimization problems are inherently multimodal, the demand for good global optimizer is more urgent.

Particle swarm optimizer (PSO) [1] [2], inspired by the simplified social model, is a new computing technique of swarm intelligence. In the PSO, each particle locates at a position that denotes its solution in the search space. Then the particle evolves its solution by moving towards the personal and the global best positions. Different from the other computing techniques, such as genetic algorithm (GA) [3] and ant colony optimizer (ACO) [4], each particle in the PSO maintains a direct learning from the best solution found so far. In other words, the best solution leads the evolution of the swarm. This is the reason for the fast convergence speed of the PSO, but it also reduces the diversity of the swarm and makes the PSO vulnerable to local optima.

The PSO is distinguished for its fast convergence speed and easy implementation, and it has been applied in various fields, such as the optimization of power system [5], the assignment problem [6], and design for antennas [7], etc. However, the traditional PSO usually has difficulty in solving multimodal solutions [8]-[10]. In order to conquer this weakness, a PSO with lifespan (LS-PSO) is proposed. Since all the particles learn from the best-so-far solution, it can be considered as the leader of the swarm. In the LS-PSO, the lifespan of a leader is a function of its power to lead the swarm towards better solutions. The stronger its leading power is, the longer its lifespan will be. When its lifespan is exhausted, the leader will be replaced by a new solution with stronger leading power and an initial lifespan. In essence, the proposed LS-PSO changes the definition of the leader from the best-so-far solution to the solution with the greatest leading power. Our paper provides a first attempt under such a redefinition, therefore the details of the LS-PSO still wait for further discussions and improvements. However, the experiments on six multimodal functions of thirty dimensions have already revealed its power. The numerical results shows that the LS-PSO comprehensively outperforms the traditionally PSO.

The remainder of this paper is organized as follows. Section II introduces the traditional PSO and reviews some previous related work. Section III details the proposed LS-PSO. Section IV displays the settings and results of the experiments. Finally, section V draws a conclusion.

II. Brief review on PSO

A. Framework of traditional PSO

The particle swarm optimizer (PSO) [1] [2] was inspired by the social behaviors in nature. In a *D* dimensional search space, X_i and v_i $(i = 1,2,\cdots,N)$ are *D*-dimensional vectors denoting the position and the velocity of the *i*-th particle, respectively. $pbest_i$ (i=1,2,…,N) and *gbest* represent the best solutions that are ever found by the *i*-th particle and the whole swarm, respectively. In every iteration, the PSO updates the velocity and the position of one particle as the following equations.

$$\begin{aligned} v_i &= w \times v_i \\ &+ c_1 \times rand_1 \times (pbest_i - X_i) \\ &+ c_2 \times rand_2 \times (gbest - X_i), \quad i = 1,2,\cdots,N \end{aligned} \tag{2}$$

This work was supported by NSF of China Project No.60573066 and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, P.R. China.

Authors are with Department of Computer Science, SUN Yat-sen University, Guangzhou, P.R.China, (Jun Zhang is the corresponding author, email:junzhang@ieee.org)

978-1-4244-1823-7/08/$25.00©2008 IEEE

$$X_i = X_i + v_i, \qquad i = 1,2,\cdots,N, \tag{3}$$

The operators in (3) and (4) are operators of vectors, implying that the same operations are performed on each dimension of the related vector. For example, the notation '+' in (3) means to do the adding operation on each dimension of X_i and v_i. The w in (2) is the scalar of inertia weight [11], which defines proportion to learn from the previous velocity. c_1 and c_2 are scalars that determine the weight to learn from $pbest_i$ and *gbest*, respectively. $rand_1$ and $rand_2$ are the craziness vectors [1]. Note that the variables of $rand_1$ and $rand_2$ can be either the same or different on each dimension. A flowchart of the traditional PSO is displayed in Fig.1.

As can be seen from (2), in every iteration every particle learns from *gbest*, thus the role of *gbest* immediately follows: the leader of the swarm during the evolution. When *gbest* falls on a local optimum, there is great possibility that the whole swarm is trapped in the permaturity.

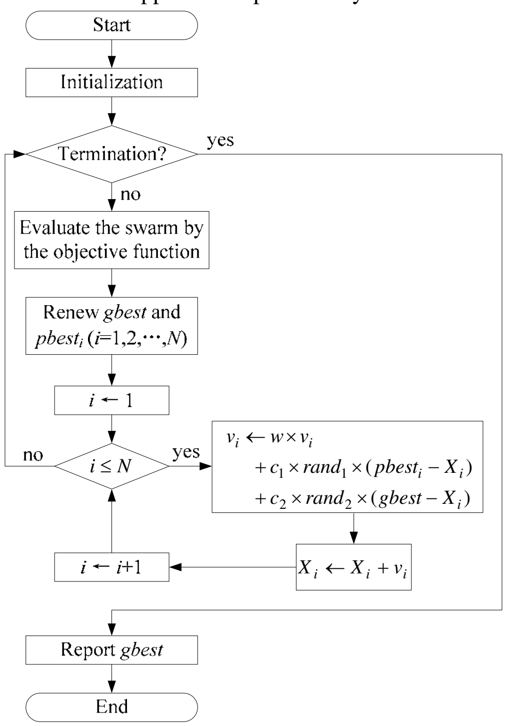


Fig.1. Flowchart of the traditional PSO

B. Related work

Ever since its first proposal in 1995 [1] [2], the PSO has developed various interesting variants. Many of them are focused on improving the performance of the PSO as a global optimizer. Here we choose some typical ones and classify them into five categories.

The early improvements of PSO mainly concerned its parameter settings. In 1998, Shi and Eberhart [11] introduced a new parameter, which is the inertia weight w in (2), into the PSO. The inertia weight controls the degree of learning from the previous velocity. The diversity of the swarm rises when the inertia weight increases [11]. Until now, the inertia weight is used in almost all the algorithms derived from PSO, and it has gradually become a part of the traditional algorithm. Shi and Eberhart also suggested the time decreasing [11] and fuzzy adaptive inertia weight [12], which tried to balance the exploration and exploitation dynamically.

In the second category, some inhomogeneous operators, such as the selection [13], crossover [14], and mutation [15] operators of GA, are combined with the PSO. These operators bring some improvements in solution quality. However, they usually take more computational time [16].

The third category is relatively new, in which the cooperative PSO with multiple swarms is investigated. Considering that the PSO works better in low-dimensional space, Bergh and Engelbrecht [17] decomposed the original search space and utilized multiple swarms to optimize different components of a solution vector cooperatively. Though problems like pseudominima still exist, the cooperative PSO has provided a distributed way to improve the global searching ability and obtained encouraging results. Zhang *et al*. [18] had multiple sub-swarms to compete in the same district of the search space. The winner stays and exploits for solutions more precise, while the losers are obliged to leave and explore other districts. The experiments on four benchmark multimodal functions showed the advantages of the multi-sub-swarm PSO.

The variant PSOs in the forth category use different measures to avoid the concentration of the particles. For instance, Lovbjerg and Krink [19] calculated the criticality of a particle in every iteration. If the criticality exceeded a predefined threshold, the particle is relocated. Besides, Xie *et al*. [20] proposed to use the negative entropy to avoid collisions of the particles. The objective for these schemas is to increase the diversity of the swarm, thus the PSO can escape from local optima.

The fifth category contains all the variants that decide to learn from solutions besides *gbest* and $pbest_i$ (i=1,2,…,N), which makes it the largest category in five. The typical instances include the comprehensive learning PSO (CLPSO) [8], the fully informed PSO (FIPS) [9], the fitness-distance-ratio PSO (FDR-PSO) [21], and all the other PSOs that employ the notion of neighborhood [22]-[24]. The CLPSO utilizes multiple $pbest_i$ to update the velocity of one particle and thus preserves the diversity of the swarm. The FIPS suggests different topologies of the swarm, and the update formula of the velocity for one particle is associated with its adjacencies in the topology. The FDR-PSO defines neighborhoods and replaces $pbest_i$ in (2) with the solutions that yields the largest fitness-distance ratio in the corresponding neighborhood. All these variants are based on the idea that the best solution does not have to imply the region of the global optimum, thus the swarm should not only learn from *gbest* and $pbest_i$. Extensive learning can help the PSO keep exploring the unknown districts of search space.

The PSO proposed in this paper should be classified into the fifth category. It provides a new definition for the leader of the swarm, thus the particles not only learn from *gbest* but also some fresh solutions with great leading power. The redefinition of the leader is supposed to help the PSO escape from prematurity, and consequently, help the PSO achieve good performances on multimodal problems.

III. PSO WITH LIFESPAN (LS-PSO)

A. Notion of lifespan

As discussed in Section II, *gbest* seriously influences the evolution of the swarm. In a sense, its role in the PSO is very similar with the leader of the social animals in nature. In the natural world, all kinds of leaders have lifespan in proportion to their power to lead the community to survival and prosperity. If their leading power declines, new leaders with greater leading power and different knowledge will be elected and replace the old ones. The proposed PSO with lifespan (LS-PSO) transplants the notion of lifespan from the natural leader to the leader of the swarm in the PSO. In other words, the leader in the traditional PSO, *gbest*, is endowed with a lifespan to determine its valid time. The LS-PSO adjusts the lifespan of *gbest* adaptively according to its power to lead the swarm towards better solutions. When the original *gbest* has run out of its lifespan, a more influential solution, which is composed of materials different from the original one, is generated for replacement.

To realize the LS-PSO, there are two key components. One is the lifespan controller, which determines the way to adjust the lifespan of *gbest*. The other one is the producer of new leaders, which forms new solutions with greater leading power and inserts them into the swarm. Both of these components are based on the standard of evaluating the leading power of a solution. The following paragraphs will detail the realization of these two components. The descriptions are made in the assumption that the optimized object is a maximum problem.

B. Lifespan controller

The essence of the lifespan controller is to adjust the lifespan of *gbest* in proportion to its leading power. If *gbest* can lead the swarm to discover better solutions, the controller increases its lifespan. On the contrary, the controller reduces the lifespan of *gbest* when it fails to improve the swarm. Though the essence is the same, the lifespan controller varies when the ways to evaluate the leading power are different. This paper introduces a lifespan controller based on the relationship between $pbest_i$ and *gbest*.

Supposing $gbest(k)$ and $pbest_i(k)$ are *gbest* and $pbest_i$ in the k-th iteration, they satisfy

$$gbest(k) = \arg\max_{i\in\{1,2,\cdots,N\}}\{f[pbest_i(k)]\}\,. \quad (4)$$

Moreover, both $gbest(k)$ and $pbest_i(k)$ are sequences in non-descending order, which has

$$\begin{aligned}\delta_{gbest}(k) &= f[gbest(k)] - f[gbest(k-1)] \ge 0\\ \delta_{pbest_i}(k) &= f[pbest_i(k)] - f[pbest_i(k-1)] \ge 0\end{aligned},\quad i = 1,2,\cdots,N$$

$$. \quad (5)$$

A rule as (6) can be concluded according to (4) and (5), which means that the improvement of *gbest* promises the improvement of $pbest_i$ (i=1,2,…,N).

$$\begin{aligned}&\delta_{gbest}(k) > 0\\ &\Rightarrow \exists\, i \in \{1,2,\cdots,N\},\ \text{ such that } \delta_{pbest_i}(k) > 0 \quad (6)\\ &\Rightarrow \textstyle\sum_{i=1}^{N}\delta_{pbest_i}(k) > 0\end{aligned}$$

However, the improvement in $\sum_{i=1}^{N}\delta_{pbest_i}(k) > 0$ can not promise the improvement of $\delta_{gbest}(k) > 0$.

Based on (6), the situations during the evolution of the PSO can be classified into the following three cases.

Case 1) $\delta_{gbest}(k) > 0$, implying $\sum_{i=1}^{N}\delta_{pbest_i}(k) > 0$. *gbest* is able to lead the swarm to find better solution, which makes the controller increase its lifespan.

Case 2) $\delta_{gbest}(k) = 0$ but $\sum_{i=1}^{N}\delta_{pbest_i}(k) > 0$. The general quality of the swarm is improved by learning to *gbest*, but the swarm fails to find better solutions. This case implies the influence of *gbest* is weakening. Thereby the lifespan of *gbest* is reduced by σ_1^- and a more influential *gbest* is expected.

Case 3) Both $\delta_{gbest}(k)$ and $\sum_{i=1}^{N}\delta_{pbest_i}(k)$ equal zero. In this case, the swarm has great possibility to be trapped in a local optimum. The leading power of *gbest* is almost lost. Thus the controller subtracts $\sigma_2^-(\sigma_2^- > \sigma_1^-)$ from the lifespan, which accelerates the birth of a new *gbest*.

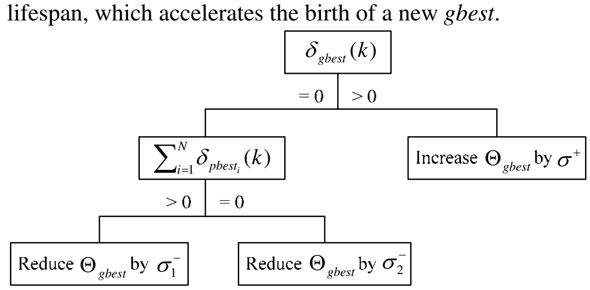


Fig.2. A tree structure of the lifespan controller. Θ_{gbest} denotes the lifespan of *gbest*.

Fig.2 summarizes the above three rules in a structure of tree, in which Θ_{gbest} denotes the lifespan of the current *gbest*. The lifespan controller is important. It determines the timing of employing a new *gbest*, and thereby adjusts the proportion of exploration to exploitation. A good balance between and

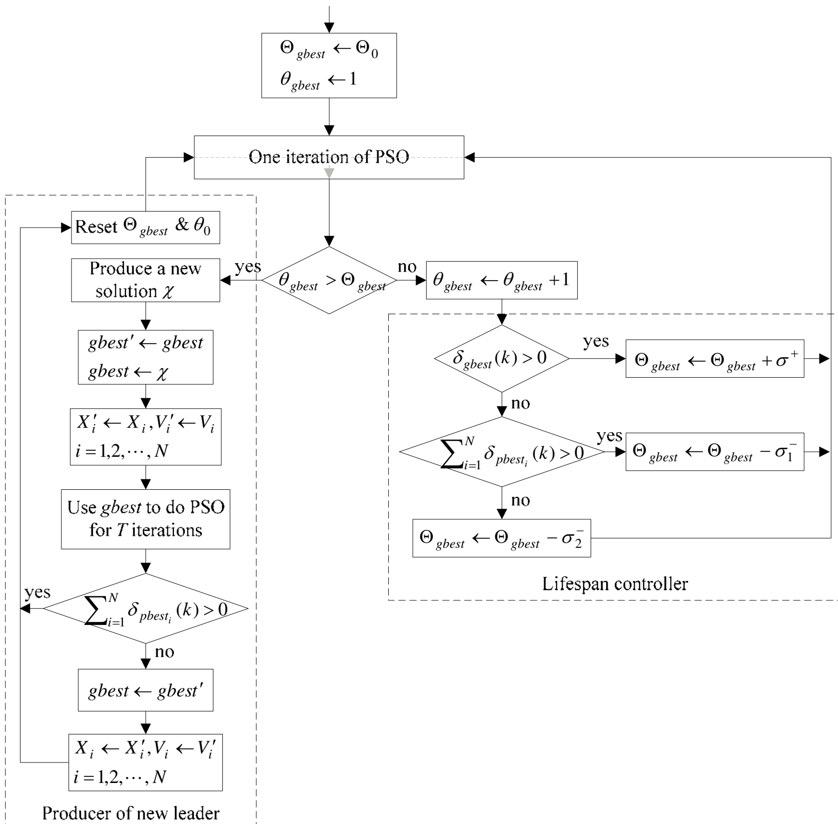


Fig.3. The flowchart of the combination of the two key components in the LS-PSO.

exploitation can keep the PSO from prematurity.

A. Producer of new leaders

The solution produced here is supposed to stay away from *gbest*, so that the swarm can attempt at evolving in different directions. The producer generates a new solution by replacing $gbest^j$ with a stochastic number within the legal range $[L^j, U^j]$ (j=1,2,…,D). The probability of the replacement on the j-th dimension is decided by a predefine constant $\mu \in (0,1)$. Then the new solution $\chi = (\chi^1, \chi^2, \cdots, \chi^D)$ can be described as

$$\chi^j = \begin{cases} random(L^j, U^j) & rand^j < \mu \\ gbest^j & \text{otherwise} \end{cases}, j = 1,2,\cdots,D, \qquad (7)$$

where $gbest^j$ is the variables of the vector *gbest*. The producer tests the leading power of χ by using it as the leader for *T* iterations. After *T* iterations, if

$$\sum_{i=1}^{N} \delta_{pbest_i}(k+T) = \sum_{i=1}^{N} [pbest_i(k+T) - pbest_i(k)] > 0, \qquad (8)$$

χ is accepted as the new position of *gbest*. Otherwise, *gbest* remains unchanged and the particles are rolled back to the status before *T* iterations.

Note that there is a legal range $[\theta_{\min}, \theta_{\max}]$ for the value of θ_{gbest}. The producer will only be called when $\theta_{gbest} > \theta_{\min}$, which prevent over-frequent attempt to generate new leaders. However, when $\theta_{gbest} > \theta_{\max}$, the LS-PSO is forced to call the producer to prevent a long-life leader from trapping the algorithm.

B. Complete LS-PSO

The LS-PSO is erected by combining the lifespan controller and the producer of new leader. At the beginning, Θ_{gbest} is initialized to a predefined constant Θ_0, and the age of *gbest* is set as $\theta_{gbest} = 1$. In every iteration, θ_{gbest} is increased by 1 and Θ_{gbest} is adjusted by the lifespan controller. When $\theta_{gbest} > \Theta_{gbest}$, the current *gbest* has exhausted its lifespan and the producer is called to provide a new solution. Supposing the new solution is accepted, θ_{gbest} and Θ_{gbest} are reset as their initial values. The situation becomes more complex when the new solution is rejected. Generally speaking, the reset is done in such a way that the gap between θ_{gbest} and Θ_{gbest} is in proportion to the previous value of θ_{gbest}.

Fig.3 displays the combination of the lifespan controller and the producer of new leaders. In the process of LS-PSO, the controller and the producer are positioned right before the update of the velocities.

IV. EXPERIMENTS AND DISCUSSIONS

A. Parameter settings

The parameters of the original PSO are set as follows: c_1=c_2=2, w=0.5, and N=10. Besides, in order to prevent the swarm from overspeed, the maximum velocity is defined as

$$v_{\max}^j = k \cdot (U^j - L^j),\ j = 1,2,\cdots,D \tag{9}$$

and the velocity on every dimension is limited to the range of $[-v_{\max}^j, v_{\max}^j]$. Here k is set at 0.5, which is a relatively large value. Supposing the new *gbest* produced in the LS-PSO is far from the swarm, a large k enables the swarm to fly over the distance and fall around the new *gbest*.

TABLE I TEST FUNCTIONS

No.	ε	Equations	$[L^j, U^j]^D$ (j=1,2,…,D)
1	0.25	$f_1 = 418.9829 \times D - \sum_{j=1}^{D} x^j \sin(\sqrt{\lvert x^j \rvert})$	$[-500,500]^{30}$
2	0.01	$f_2 = \sum_{j=1}^{D} [(x^j)^2 - 10\cos(2\pi x_j) + 10]$	$[-5.12,5.12]^{30}$
3	0.01	$f_3 = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{j=1}^{D}(x^j)^2}\right) - \exp(\frac{1}{D}\sum_{j=1}^{D}\cos(2\pi x^j) + 20 + e$	$[-32,32]^{30}$
4	0.01	$f_4 = \frac{1}{4000}\sum_{j=1}^{D}(x^j)^2 - \prod_{j=1}^{D}\cos\left(\frac{x^j}{\sqrt{j}}\right) + 1$	$[-600,600]^{30}$
5	0.01	$f_5 = \frac{\pi}{D}\left\{ 10\sin^2(\pi y^j) + \sum_{j=1}^{D-1}(y^j - 1)^2[1 + 10\sin^2(\pi y^{j+1})] + (y^D - 1)^2 \right\}^{*} + \sum_{i=1}^{D} u(x^j, 10, 100, 4)$	$[-50,50]^{30}$
6	0.01	$f_6 = \frac{1}{10}\left\{ 10\sin^2(3\pi x^1) + \sum_{j=1}^{D-1}(x^j - 1)^2[1 + sih^2(3\pi x^{j+1})] + (x^D - 1)^2\left(1 + \sin^2(2\pi x^D)\right) \right\}^{*} + \sum_{i=1}^{D} u(x^j, 5, 100, 4)$	$[-50,50]^{30}$

* $y^j = 1 + \frac{1}{4}(x^j + 1)$, $u(x^j, a, k, m) = \begin{cases} k(x^j - a)^m & x^j > a \\ 0 & -a \le x^j \le a \\ k(-x^j - 1)^m & x^j < -a \end{cases}$

The parameters involved in the proposed LS-PSO includes the initial lifespan Θ_0, the increment and decrements in the controller σ^+, σ_1^-, and σ_2^-, the number of the testing iteration T, the reasonable range of age $[\theta_{\min}, \theta_{\max}]$, and the changing probability μ in the producer. The best values of these parameters may vary on different problems. According to experiments, one of their good combinations is $\Theta_0 = 30$, $\sigma^+ = 5$, $\sigma_1^- = 1$, $\sigma_2^- = 5$, T=2, $\theta_{\min} = 15$, $\theta_{\max} = 100$, and $\mu = 1/D$.

This section is going to compare the performances of the LS-PSO with the traditional PSO and the PSO with a static lifespan (SLS_PSO). The common parameters in these three algorithms are set the same. Each algorithm is run for 30 times and the termination in each trial is determined by the maximum evaluations, which equals to 200,000.

B. Test functions

This paper is focused on improving the global searching ability of PSO, thereby no unimodal function but six multimodal functions of 30 dimensions are used for experiments. All the functions, which are minimum problems with the optima at zero, are listed in Table I.

In Table I, the notation ε stands for the upper bound of the absolute error. One trial succeeds when the result found in this trial satisfies the bound. $[L^j, U^j]^D$ defines the solution space. Note that except for f_1, all the functions find their global optima at the centre of their solution spaces. Therefore, biased initializations are applied on f_2 to f_6, which means x^j is initialized to a random value within the asymmetric range of $[L^j, U^j/2]$.

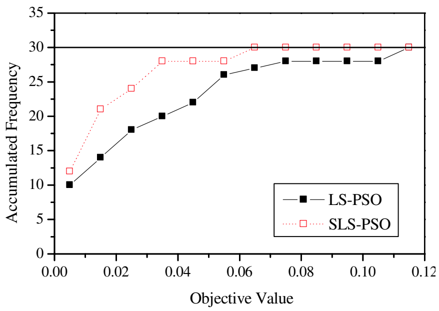
C. LS-PSO on multimodal functions

Table II shows the results of the algorithms in comparison. Each algorithm takes two columns in the table. The first column presents the statistic values. From top down, the items are the best, the worst, the median and the average results in 30 runs. The second column displays the successful rate in accordance with ε. The best values in the same items are bold.

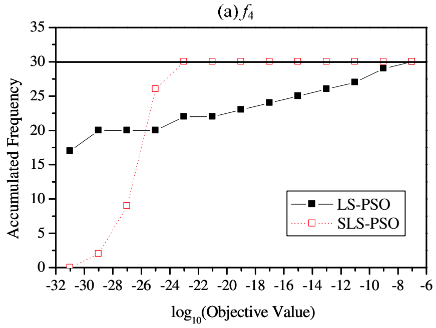
TABLE II RESULTS OF TRADITIONAL PSO, SLS-PSO, AND LS-PSO

f	Items	Traditional PSO		SLS-PSO		LS-PSO	
		Values	OK(%)	Values	OK(%)	Values	OK(%)
1	best	3691.1	0	**0.000382**	**100**	**0.000382**	**100**
	median	4687.6		**0.000382**		**0.000382**	
	worst	5515.3		0.004752		**0.000382**	
	mean	*4676.8*		*0.000535*		***0.000382***	
2	best	39.798	0	7.11×10^{-15}	96.7	**0**	**100**
	median	85.566		7.48×10^{-8}		**0**	
	worst	159.19		0.996		**3.55×10^{-15}**	
	mean	*94.189*		*0.033*		***3.55×10^{-16}***	
3	best	1.12×10^{-14}	6.7	1.15×10^{-11}	**100**	**1.12×10^{-14}**	**100**
	median	2.887		1.42×10^{-10}		**3.61×10^{-14}**	
	worst	18.464		9.08×10^{-10}		**5.00×10^{-10}**	
	mean	*4.251*		*2.02×10^{-10}*		***2.74×10^{-11}***	
4	best	**0**	30	**0**	**40**	**0**	**40**
	median	0.027		**0.012**		0.015	
	worst	90.328		**0.061**		0.071	
	mean	*9.055*		***0.016***		*0.021*	
5	best	2.09×10^{-32}	36.7	8.95×10^{-29}	**100**	**1.57×10^{-32}**	**100**
	median	0.207		2.09×10^{-26}		**1.57×10^{-32}**	
	worst	2.802		**7.28×10^{-23}**		9.47×10^{-14}	
	mean	*0.548*		***2.78×10^{-24}***		*3.09×10^{-15}*	
6	best	1.84×10^{-31}	10	1.15×10^{-26}	**100**	**1.35×10^{-31}**	**100**
	median	0.100		3.56×10^{-24}		**1.61×10^{-31}**	
	worst	2.121		**1.13×10^{-17}**		1.15×10^{-12}	
	mean	*0.354*		***3.77×10^{-19}***		*4.70×10^{-14}*	

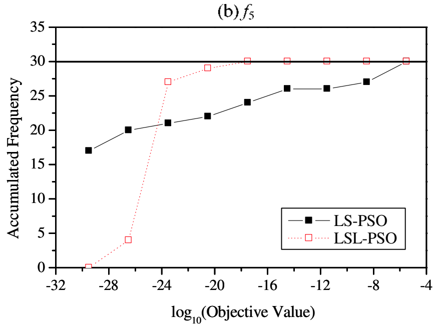
As can be seen, the average results of traditional PSO are worst than the other two algorithms on all the test functions. The LS-PSO has comprehensive advantages over the tradition PSO in all the items, but the best results of SLS-PSO on f_3, f_5, and f_6 is not as good as the traditional PSO. This is because SLS-PSO calls the producer of new *gbest* without judging the need of the algorithm, which wastes the computational time. The traditional PSO is easily trapped in local optima, but it can accidentally find global optima. Thus in general, the LS-PSO and the SLS-PSO enhance the ability of global searching for PSO, but the static lifespan in the SLS-PSO weakens its performances by improper timing of new *gbest*.



(a) f_4



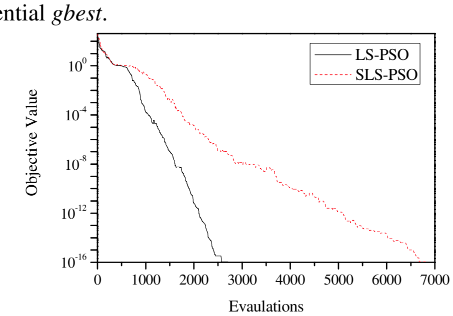
(b) f_5



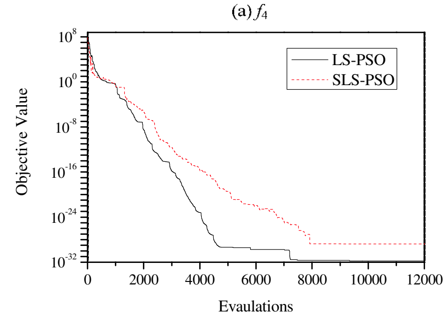
(c) f_6

Fig.4. The graphs of accumulation frequency on f_4, f_5, and f_6. The solid and the dash lines represent the LS-PSO and the PSO with static lifespan, respectively.

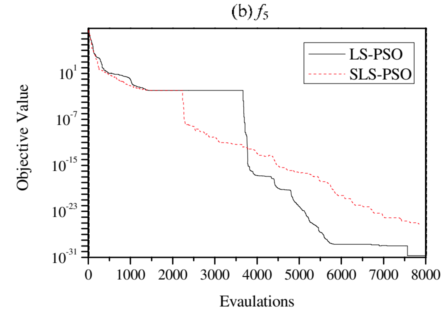
Now we will take a closer look at the comparison between the LS-PSO and SLS-PSO, where situations are divided. From f_1 to f_3, the advantages of LS-PSO are obvious. For example, the worst result of the LS-PSO is still better than the best one of the SLS-PSO on f_2. However, things change from f_4. Though the LS-PSO still beats the other one on the best and the median results, the SLS-PSO obtains better values in term of the worst and the average results. For further analysis, the curves of the accumulation frequency about the results from both algorithms are compared in Fig.4. Since the results of f_5 and f_6 are very small, the common logarithm of the results is used to describe the difference among the values. As can be seen, the solid curve of LS-PSO approaches the horizontal line at 30 gradually, but the dash line rises dramatically and reaches 30 earlier than the LS-PSO. This phenomenon indicates that the results of LS-PSO are more disperse than that of SLS-PSO. Thus even though the LS-PSO can achieves near-optima results in most trials, its failures in the other trials weaken its general performance. For example, the LS-PSO finds 17 results in [10^{-32}, 10^{-30}] out of 30 trials on f_5, but it is trapped around 10^{-10}, too. The results of the SLS-PSO has a zero frequency in [10^{-32}, 10^{-30}], but it still beats the LS-PSO because of its concentration in [10^{-28}, 10^{-24}]. Therefore, it should be said that the LS-PSO is better at locating global optima, but its performances are not stable. This weakness is most likely to result from the improper reset of θ_{gbest} and Θ_{gbest} after the failure to find a more influential *gbest*.



(a) f_4



(b) f_5



(c) f_6

Fig.5. The convergence graphs on f_4, f_5, and f_6 for both of LS-PSO and SLS-PSO. The solid and the dash lines denote the LS-PSO and the SLS-PSO, respectively.

Fig.5 selects the trials that obtain the best results to compare the convergence processes of both of LS-PSO and

SLS-PSO on f_4 to f_6. It can be seen that the solid line of LS-PSO inclines more than the dash line of SLS-PSO at most of the time, which means the LS-PSO converges faster. Besides, the solid line generally lies below the dash lines, implying the best results found by LS-PSO are better.

In general, the LS-PSO improves the power of exploration for the traditional PSO. Since the new *gbest* is always provided at the proper time, the LS-PSO can on one hand avoid prematurity, and on the other hand, save more computational time to enhance the precision of the results.

V. CONCLUSIONS

This paper proposes that the leader of the swarm in the PSO should not only be determined by its objective value but also its power to lead the swarm towards better solutions. With such a notion, a PSO with lifespan (LS-PSO) is presented. The proposed LS-PSO adjusts the lifespan of the leader in accordance with its leading power. When the old leader exhausts its lifespan, a new solution is produced and conditionally replaces the original one. The LS-PSO solves the problem of prematurity. Experiments on six multimodal functions show that the LS-PSO can achieve much better results than the traditionally PSO.

This paper utilizes the relationship between $pbest_i$ and *gbest* as the standard of the leading power. In fact, this standard can be changed. For example, the standard can also be the number or the degree of the improved $pbest_i$, thus the lifespan controller will change accordantly. Besides the lifespan controller, the producer of new *gbest* also has room for improvement. The producer used in this paper is a kind of blindness, therefore we are looking for more effective methods. In a word, the future work on the LS-PSO focuses on a more precise standard for the leading power and a smarter producer for new *gbest*.

REFERENCES

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromachine Human Sci.*, Nagoya, Japan, 1995, pp. .39-43.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, 1995, pp. 1942-1948.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[4] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Massachusetts, London, MIT Press, 1997.

[5] T. O. Tin, M. V. C. Rao, and C. K. Loo, "A novel approach for unit commitment problem via an effective hybrid particle swarm optimization," *IEEE Trans. Power System*, vol. 21, pp. 411 – 418, Feb. 2006.

[6] S. Y. Ho, H. S. Lin, W. H. Liauh, and S. J. Ho, "OPSO: Orthogonal particle swarm optimization and its application to task assignment problems", *IEEE Trans. SMC*, *Part A*. vol. 38, pp. 288 – 289, Mar. 2008.

[7] S. Genovesi, A. Monorchio, R. Mittra, and G. Manara, "A sub-boundary approach for enhanced particle swarm optimization and its application to the design of artificial magnetic conductors," *IEEE Trans. Antennas and Propagation*, vol. 55, pp. 766 – 770, Mar. 2007.

[8] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particles swarm optimization for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, pp. 281-295, Jun. 2006.

[9] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 204–210, Jun. 2004.

[10] X. Chen and Y. M. Li, "A modified PSO structure resulting in high exploration ability with convergence guaranteed," *IEEE Trans. SMC Part B*, vol. 37, pp. 1271 – 1289, Oct. 2007.

[11] Y.Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput*., 1998, pp. 69-73.

[12] Y.Shi and R. C. Eberhart, "Particle swarm ptimization with fuzzy adaptive inertia weight," in *Proc. Workshop Particle Swarm Optimization*, Indianapolis, IN, 2001, pp. 101–106.

[13] P. J. Angeline, "Using selection to improve particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, Anchorage, AK, 1998, pp. 84–89.

[14] M. Lovbjerg, T. K. Rasmussen, and T. Krink, "Hybrid particle swarm optimizer with breeding and subpopulations," in *Proc. Genetic Evol. Comput. Conf.*, 2001, pp. 469–476.

[15] R. A. Krohling, "Gaussian particle swarm with jumps," in *Proc. IEEE Congr. Evol. Comput.*, 2005, Edinburgh, Scotland, UK, pp. 1226-1231.

[16] J. H. Seo, C. H. Im, C. G. Heo, J. K. Kim, H. K. Jun, and C. G. Lee, "Multimodal function optimization based on particle swarm optimization," in *IEEE Trans. Magnetics*, vol. 42, pp. 1095-1098, Apr. 2006.

[17] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 225–239, Jun. 2004.

[18] J. Zhang, D. S. Huang, K. H. Liu, "Multi-sub-swarm particle swarm optimization algorithm for multimodal function optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2007, Singapore, pp. 3215-3220.

[19] M. Lovbjerg and T. Krink, "Extending particle swarm optimizers with self-organized criticality," in *Proc. Congr. Evol. Comput.*, Honolulu, HI, 2002, pp. 1588–1593.

[20] X. Xie, W. Zhang, and Z. Yang, "A dissipative particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Honolulu, HI, 2002, pp. 1456–1461.

[21] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-distance-ratio based particle swarm optimization," in *Proc. Swarm Intelligence Symp.*, 2003, pp. 174–181.

[22] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1931–1938.

[23] P. N. Suganthan, "Particle swarm optimizer with neighborhood operator," in *Proc. Congr. Evol. Comput.*, Washington, DC, 1999, pp. 1958–1962.

[24] X. Hu and R. C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proc. Congr. Evol. Comput.*, Honolulu, HI, 2002, pp. 1677–1681.