改进的粒子群优化算法的研究和分析

田东平 [1,2],徐成虎 [3]
TIAN Dong-ping[1,2],XU Cheng-hu[3]

1.宝鸡文理学院 计算机软件研究所,陕西 宝鸡 721007
2.宝鸡文理学院 计算机科学系,陕西 宝鸡 721007
3.宝鸡文理学院 教育科学系,陕西 宝鸡 721007
1.Institute of Computer Software,Baoji University of Arts and Science,Baoji,Shaanxi 721007,China
2.Department of Computer Science,Baoji University of Arts and Science,Baoji,Shaanxi 721007,China
3.Department of Education Science,Baoji University of Arts and Science,Baoji,Shaanxi 721007,China
E-mail:tdp211@yahoo.com.cn

TIAN Dong-ping,XU Cheng-hu.Study and analysis of improved Particle Swarm Optimization algorithm.Computer Engineering and Applications,2008,44(34):56-60.

Abstract: Particle Swarm Optimization (PSO) is a novel stochastic global optimization evolutionary algorithm.To efficiently control the global search and local search of PSO and obtain a better balance between them.In this paper,a new particle swarm optimization is proposed based on the threshold of evolutionary generation,maximal focusing distance and Gaussian mutation among particles.The new algorithm includes Gaussian mutation operator during the running time,which,through the quantization decision of particles focusing degrees,can be very useful to improve the ability of PSO in breaking away from the local optimum.The experimental results show that the proposed algorithm is feasible and effective.

Key words: Particle Swarm Optimization(PSO);global search;local search;maximal focusing distance;Gaussian mutation

摘 要: 粒子群优化算法是一种新的随机全局优化进化算法。为了有效地控制其全局搜索和局部搜索,使之获得较好的平衡,论文在深入分析和研究标准粒子群优化算法的基础上,提出了一种基于进化代数阈值和粒子间最大聚集距离高斯变异的粒子群优化算法。该算法在运行过程中通过粒子聚集程度的量化判定,对当前的最优粒子施加高斯变异,从而增强粒子群优化算法跳出局部最优解的能力。测试函数仿真结果表明了该算法的可行性和有效性。

关键词: 粒子群优化算法;全局搜索;局部搜索;最大聚集距离;高斯变异

DOI:10.3778/j.issn.1002-8331.2008.34.016 **文章编号:**1002-8331(2008)34-0056-05 **文献标识码:**A **中图分类号:**TP301.6

1 引言

粒子群优化算法(Particle Swarm Optimization,PSO)是一种基于群体智能(Swarm Intelligence)的群优化算法和群搜索算法。因受到人工生命(Artificial Life)的研究结果启发,Kennedy 和 Eberhart 于 1995 年提出了粒子群优化算法[1],并立刻引起了优化及演化计算等诸多领域学者们的普遍青睐。目前,PSO 已被广泛应用于函数优化、神经网络训练、模式分类、模糊系统控制及其他的应用领域。然而,PSO 算法亦有其不足:易陷入局部极值点,进化后期收敛速度慢,精度较差等。为了克服 PSO 算法的这些不足,研究人员提出了许多改进的 PSO,如:Kennedy[2](2002)、Natsuki[3](2003)、Ratnaweera[4](2004)、Monson[5](2004)、Van[6](2004)、Rodriguez[7](2004)、Katare[8](2004)、Sun[9](2004)等,这些算法从不同角度对基本 PSO 进行了改进,在不同程度上提高了算法的收敛速度和计算精度。特别是近两年来国内有关学者提出了基于 PSO 的各种混合软计算,如:刘华蓥[10]提出的基于混沌搜索解决早熟收敛的混合 PSO 算法(COPSO)、赵志刚[11]提出的带自变异算子的 PSO 算法(PSO-MO),余炳辉[12]提出的随机摄动 PSO 算法(RP-PSO),李莉[13]给出的基于混合 PSO 算法的高维复杂函数求解(HPSO)等,这些方法均应用了混合软计算的设计思想,将一些局部搜索性能极强的优化算法引入到 PSO 中,以对其性能加以改进,从而在不同程度上获得了较为理想的优化结果。

近几年的研究和实践表明,PSO 在多维空间函数寻优、动态目标寻优等方面有着收敛速度快、解质量高、鲁棒性好等优点,既适合科学研究,又特别适合工程应用[14]。

PSO 的优势在于算法简单、易于实现,可调参数少;同时又有深刻的智能背景。然而,与其它全局优化算法一样,PSO 同样存在早熟收敛现象,尤其是在比较复杂的多峰函数优化问题中。目前解决这一问题的主要方法是增加粒子群的规模,这样做虽然对算法的性能有一定改善,但是同样存在缺陷:一是不

作者简介:田东平(1981-),男,讲师,主研方向为模糊推理、专家系统和智能优化计算;徐成虎(1980-),男,主研方向为算法设计和教育统计。
收稿日期:2007-10-10 修回日期:2008-01-28

能从根本上克服早熟收敛问题；二是会大量增加算法的运算量。为此，在深入分析和研究基本粒子群优化算法的基础上，提出了一种基于粒子间最大聚集距离高斯变异的粒子群优化算法。该算法在运行过程中通过粒子聚集程度的量化判定，对当前的最优粒子施加高斯变异，从而提高粒子群优化算法跳出局部最优解的能力。实验结果表明，在取相同的阈值代数情况下，惯性权重取定值的粒子群优化算法（Cω-PSO）、惯性权重线性减小的粒子群优化算法（LDω-PSO）、惯性权重取定值且基于粒子间最大聚集距离高斯变异的粒子群优化算法（Cω-GMPSO）、惯性权重线性减小且基于粒子间最大聚集距离高斯变异的粒子群优化算法（LDω-GMPSO）中，本文所提出策略相应算法的性能均优于其他算法。此外，论文测试结果与已有文献[10-13]进行了比较，对比结果显示：文中所提出PSO算法的优化性能，均达到或者优于文献中的算法，这进一步说明了本文算法的可行性和有效性，在进行高维、多峰的纯数值函数求解时能取得较为满意的优化结果。

2 粒子群优化算法

2.1 基本PSO原理阐述

粒子群优化算法（PSO）是1995年由Kennedy和Eberhart在鸟群、鱼群和人类社会行为规律的启发下提出的一种基于群体智能（Swarm Intelligence）的演化计算技术。PSO算法模拟鸟群飞行觅食的行为，通过鸟之间的集体协作使群体达到最优目的。在PSO系统中，每个备选解被称为一个"粒子"，多个粒子共存与合作寻优。而每个粒子根据其自身"经验"和相邻粒子群的最佳"经验"在问题空间中向更好的位置"飞行"，以便搜索最优解。

PSO算法的数学表示如下[15]：

设搜索空间为D维，总粒子数为N。第i个粒子位置表示为$X_i=(x_{i1},x_{i2},\cdots,x_{iD})$；第$i$个粒子"飞行"历史中的最优位置（即个体极值）为$P_i=(p_{i1},p_{i2},\cdots,p_{iD})$；设$P_g$为所有$P_i$（$i=1,2,\cdots,N$）中的最优值（即全局极值）；第$i$个粒子的位置变化率（即速度）为向量$V_i=(v_{i1},v_{i2},\cdots,v_{iD})$。每个粒子的位置按如下公式进行变化：

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

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

式（1）和式（2）中，c_1，c_2为正常数，称为加速因子；r_1，r_2为[0，1]之间的随机数；ω称为惯性因子，ω较小适于进行小范围的开发（Exploitation），ω较大适于对解空间进行大范围的探测（Exploration）；α为非负常数，称为约束因子，目的是用于控制速度的权重；第d维（$1\leqslant d\leqslant D$）的位置变化范围为$[-XMaxd, XMaxd]$，速度变化范围为$[-VMaxd, VMaxd]$，迭代过程中若位置和速度超过边界范围则取边界值。粒子群初始位置和速度随机产生，然后按式（1）、式（2）进行迭代，直至找到满意解。

2.2 基本PSO收敛性分析

（1）在文献[16]中，Clerc对式（1）和式（2）中的参数做了初步分析，尽管存在明显缺陷，但对于研究和探讨PSO的收敛性仍具有重要的参考意义。

首先，定义$P_{id}\Leftarrow\dfrac{c_1p_{id}+c_2p_{gd}}{c_1+c_2}$，则式（1）可简化为：

$$v_{id}(t+1)=v_{id}(t)+c(p_{id}(t)-x_{id}(t)) \quad (3)$$

其中，$c=c_1+c_2$。进一步简化系统，假设p_{gd}，c不变，且解空间仅一维，研究一个粒子，则式（1）、式（2）可变为状态方程的形式：

$$\begin{cases}v(t+1)=v(t)+c(p-x(t))\\x(t+1)=x(t)+v(t+1)\end{cases} \quad (4)$$

其中，p和c是常数，通过对状态转移矩阵（4）的稳定性分析，即可得出粒子运动稳定性的约束条件。其结论如下：当式（1）和式（2）中参数满足约束条件

$$v_{id}(t+1)=K\times[v_{id}(t)+c_1\times r_1\times(p_{id}(t)-x_{id}(t))+c_2\times r_2\times(p_{gd}(t)-x_{id}(t))] \quad (5)$$

$K=\dfrac{2}{\left|2-c-\sqrt{c^2-4c}\right|}$，$c=c_1+c_2>4$时，在假设条件下粒子运动轨迹稳定。

（2）在文献[17]中，给出了粒子群收敛性与群体适应度方差之间的关系。

首先，定义粒子群的群体适应度方差：

$$\sigma^2=\frac{1}{N}\sum_{i=1}^{N}\left(\frac{f_i-f_{avg}}{f}\right)^2 \quad (6)$$

其中，N为粒子群的粒子数目；f_i为第i个粒子的适应度；f_{avg}为粒子群目前的平均适应度；f为归一化定标因子，其作用是限制σ^2的大小。f一般取值如下：

$$f=\max\{1,\max\{|f_i-f_{avg}|\}\},i\in[1,N] \quad (7)$$

上述群体适应度方差的定义表明：σ^2反映的是粒子群中所有粒子的"收敛"程度。σ^2越小，粒子群趋于收敛；反之，粒子群处于随机搜索阶段。其次，定义粒子群中粒子的收敛：

$$\lim_{t\to+\infty}X(t)=P \quad (8)$$

其中，$X(t)$为粒子群中某个粒子在t时刻的位置，P为搜索空间内的任意位置。该定义表明：粒子的收敛是指粒子最终停留在搜索空间内某一固定位置P。

文献[18]通过严格的数学推导，得出如下结论：

$$\lim_{t\to+\infty}X(t)=\alpha\times pBest+(1-\alpha)\times pBest \quad (9)$$

其中，$\alpha=\dfrac{c_1}{c_1+c_2}$（$c_1$，$c_2$为式（1）中的加速因子）；$pBest$表示粒子当前的个体极值；$gBest$表示粒子群当前的全局极值。在此基础上，文献[5]进一步证明，如果粒子群优化算法陷入早熟收敛或者达到全局收敛，粒子群中的粒子将聚集在搜索空间中一个或几个特定的位置，且群体适应度方差σ^2等于零。

（3）在文献[19]中，给出了惯性因子与粒子速度之间的关系，并分析了它对PSO算法收敛性的影响。

首先，不妨设自变量空间为一维，群体只有一个粒子，此时则有$p_i=p_g$。又令$\phi_1=r_1c_1$，$\phi_2=r_2c_2$，$\phi=\phi_1+\phi_2$，记$y_i=p_i-x_i$，则式（1）、式（2）可简化为：

$$v(t+1)=\omega v(t)+\phi y(t) \quad (10)$$

$$y(t+1)=-\omega v(t)+(1-\phi)y(t) \quad (11)$$

对式（10）、（11）进行迭代消去y可得：

$$v(t+2)+(\phi-1-\omega)v(t+1)+\omega v(t)=0 \quad (12)$$

将式（12）连续化，可导出二阶微分方程：

$$\frac{\partial^2v}{\partial t^2}+In(e_1e_2)\frac{\partial v}{\partial t}+In(e_1)In(e_2)v=0 \quad (13)$$

其中e_1和e_2为二次方程（14）的根。

$$\lambda^2+(\phi-1-\omega)\lambda+\omega=0 \quad (14)$$

即 $e_1=(\omega+1-\phi+\sqrt{\Delta})/2$，$e_2=(\omega+1-\phi-\sqrt{\Delta})/2$，式中 $\Delta=(\omega+1-\phi)^2-4\omega$。故微分方程（13）的通解为：

$$v(t)=k_1e_1^t+k_2e_2^t \tag{15}$$

于是可进一步推出：

$$y(t)=[k_1e_1^t(e_1-\omega)+k_2e_2^t(e_2-\omega)]/\phi \tag{16}$$

其中 $k_1=(-\phi y(0)-(\omega-e_2)v(0))/(e_2-e_1)$，$k_2=(\phi y(0)+(\omega-e_1)v(0))/(e_2-e_1)$。

由式（15）、式（16）可以看出：在 $t\to\infty$，$\max(|e_1|,|e_2|)<1$ 时，有 $v(t)$，$y(t)\to 0$ 即算法收敛。若 $\omega>0.5\phi-1$，则有 $\max(|e_1|,|e_2|)<1$；而在 $\omega>0.5(c_1+c_2)-1$ 时，必有 $\omega>0.5\phi-1$ 成立，故算法一定收敛。

3　改进的粒子群优化算法

3.1　改进 PSO 设计思想

分析上述式（1）和式（2）可知，如果算法在多次迭代过程中全局最优粒子位置不发生变化，则其速度更新主要取决于局部最优粒子的变化，于是速度将越来越小，粒子群表现出强烈的“趋同性”，体现在优化性能上就是收敛速度快，但易陷入局部最优点。文献[18]提出了 Multi-start PSO，即每迭代若干次后，保留历史最优粒子，其余粒子全部初始化，以提高粒子的多样性，扩大搜索空间，摆脱局部最优点的吸引，保证算法收敛到全局最优。但是粒子群的全部初始化将完全破坏当前粒子的结构，使得收敛速度和搜索精度都大大降低。

从式（2）易知，粒子下一时刻的位置由当前位置与当前速度共同决定，速度大小决定粒子移动的距离，速度方向决定粒子前进的方向。而又由式（1）可知，粒子当前速度由三个因素决定：原来速度 $v_{id}(t)$、个体极值 $p_{id}(t)$ 和全局极值 $p_{gd}(t)$。如果算法出现早熟收敛，则全局极值 $p_{gd}(t)$ 一定是局部最优解。此时若改变 $p_{gd}(t)$，就可以改变粒子的前进方向，从而让粒子进入解空间的其它区域进行搜索，如此以来，在其后的搜索过程中，算法就有可能找到全局最优解。

基于上述分析，在 PSO 迭代过程中，增加了进化代数阈值（GT）和基于最大聚集距离（MaxDist）高斯变异（GM）的策略，以此使算法摆脱局部最优点的束缚，提高其全局搜索能力，同时又具有较快的收敛速度。最大聚集距离（MaxDist）定义如下[20]：

$$MaxDist=\max_{i=1\cdots m}\left(\sqrt{\sum_{d=1}^{D}(p_{ld}-x_{id})^2}\right) \tag{17}$$

式（17）中，m 为相邻子群粒子数；p_{ld} 为历史最佳位置；x_{id} 表示第 i 个粒子的第 d 维分量。

3.2　改进 PSO 实现流程

步骤 1　初始化粒子群中的所有粒子，在允许范围内随机设置粒子的初始位置 X_i 与初始速度 V_i，$i\in[1,N]$，N 为粒子个数；

步骤 2　将第 i 个粒子的 *pBest* 设置为该粒子的当前位置，*gBest* 设置为初始群体中最佳粒子的位置；

步骤 3　对于粒子群中的所有粒子，执行如下操作：

（1）根据式（1）和式（2）更新速度和位置；

（2）计算每个粒子的适应度 $F(X)$，$i\in[1,N]$；

（3）如果粒子 i 的适应度优于自身 *pBest* 的适应度，*pBest* 更新为该粒子的当前位置 X_i；

（4）如果粒子 i 的适应度优于 *gBest* 的适应度，*gBest* 更新为该粒子的当前位置 X_i；

步骤 4　根据式（17）计算粒子群的最大聚集距离 *MaxDist*；

步骤 5　判断迭代次数是否大于预设阈值代数，若是则执行步骤 6；否则转步骤 3；

步骤 6　判断粒子群最大聚集距离 *MaxDist* 是否小于预设聚集距离阈值，若是则执行步骤 7；否则转步骤 3；

步骤 7　对当前粒子群中全局极值位置 P_{gd} 增加随机扰动 $P_{gd}=P_{gd}\times(1+\eta)$。其中 η 是服从高斯分布的随机变量，即 $\eta\sim Gauss(0,1)$；

步骤 8　若算法达到最大迭代次数 *MaxGen*，则输出 P_{gd} 及相应目标函数的值，算法运行结束；否则转步骤 3。

4　数值仿真与分析

4.1　仿真函数引用

本文采用经典函数 Sphere 和 Rastrigrin 来测试 Cω-PSO、LDω-PSO、Cω-GMPSO 和 LDω-GMPSO 算法的优化性能。具体函数引用如下：

（1）Sphere Function　$f_1=\sum_{i=1}^{d}x_i^2$，$x_i\in[-100,100]$

（2）Rastrigrin Function　$f_2=\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)+10]$

$x_i\in[-10,10]$

其中，Sphere 是一个单模态函数，在点（0，0，…，0）达到最小值 0；Rastrigrin 是一个自变量之间相互独立的多模态函数且存在大量局部极值点，在点（0，0，…，0）达到最小值 0。

4.2　仿真结果分析

算法参数设置如下（优化结果如表 1，表 2 所示）：惯性因子 $\omega=1$，$\omega Max=0.9$ 和 $\omega Min=0.4$；加速因子 $c_1=c_2=2$；约束因子 $\alpha=0.5$；函数维度 $d=10$；群体规模 $N=40$；最大迭代次数 $MaxGen=800$；阈值代数分别取 $MaxGen/4$、$2\times MaxGen/4$ 和 $3\times MaxGen/4$；聚集距离阈值取 0.28。算法在 MATLAB6.5 环境下实现，为衡量本文算法的优化性能，对每个函数独立运行 30 次所得结果的平均值作为最终评价指标。

从表 1 和表 2 的优化结果不难发现：（1）对于单模态函数 Sphere，Cω-PSO 的优化性能明显好于 LDω-PSO；但对于多模态函数 Rastrigrin 而言，LDω-PSO 的优化效果则更好；（2）对于 Cω-GMPSO 和 LDω-GMPSO，在相同条件下，惯性因子线性减小的 PSO 要比惯性因子取定值的 PSO 优化性能更好，且收敛速度快。这充分体现了惯性因子对粒子群优化算法收敛性能的重要影响；（3）从代数阈值（GT）方面来看，随着 GT 的增大，优化结果的数量级呈现出显著下降趋势。这一特点表明了 PSO 算法在早期收敛非常快，所以应及时让算法进行高斯随机扰动以使其跳出局部最优，获得更好的优化结果；（4）运用本文算法对 Rastrigrin 函数的测试中，优化结果均达到理论最小值 0，这也进一步说明了本文算法的正确性和有效性。需要注意的是：算法实验结果中的数值“0”表示 Matlab 精度值 1e-308（Matlab 当小于这个值时显示 0）。

函数 Sphere 和 Rastrigrin 的最大聚集距离（MaxDist）如下面图 1 和图 2 所示（限于篇幅，仅给出两函数在优化方法 LDω-

表 1　对 Sphere 函数的优化结果

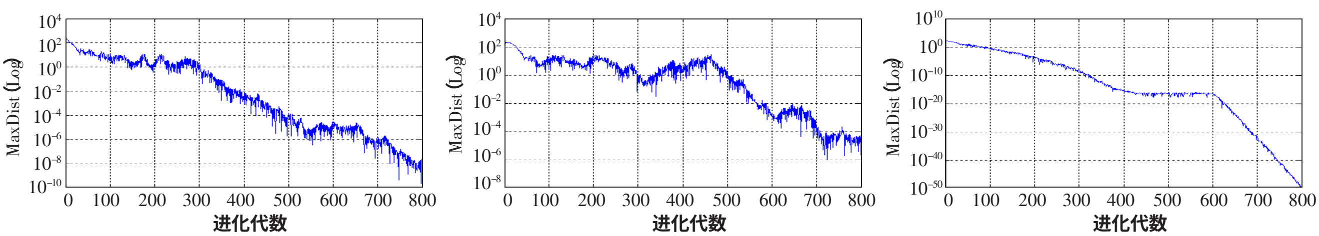
优化方法	代数阈值	平均收敛值	最优收敛值	最优收敛值对应变量				
Cω-PSO	——	0.017 5	5.938 8e-004	0.002 456 3 -0.001 127	-0.002 098 6 -0.012 116	-0.006 488 2 -0.004 556 9	-0.011 583 0.012 368	0.008 429 4 -0.003 782 6
LDω-PSO	——	0.079 1	1.003 5e-002	0.017 584 -0.018 652	-0.023 852 0.063 394	-0.059 939 0.005 994 7	0.009 761 1 -0.015 862	0.013 306 -0.025 259
Cω-GMPSO	*MaxGen*/4	7.271 2e-022	4.382 2e-026	9.925 5e-015 4.548 9e-015	6.176 7e-015 -1.464 5e-014	-1.596 4e-014 8.195 9e-016	-5.718 7e-015 2.118 9e-014	1.499 3e-014 2.188 6e-015
	2*MaxGen*/4	1.484 3e-016	6.664 3e-025	2.222 6e-013 -4.403 4e-013	-7.7133 e-014 -9.571 7e-014	1.134 6e-015 5.300 6e-014	-3.723 3e-014 8.562 5e-014	-1.387 4e-015 1.595 9e-015
	3*MaxGen*/4	2.031 2e-009	1.048 3e-019	-1.875 8e-013 -8.600 5e-013	3.122 7e-012 -7.751 4e-014	1.448e-010 6.627 5e-012	-1.133 8e-011 1.046 8e-012	-1.179 2e-012 9.725 7e-011
LDω-GMPSO	*MaxGen*/4	3.883 1e-143	3.818 1e-151	-8.217 8e-077 -1.096 4e-076	-1.252 8e-076 1.999 3e-076	3.630 5e-077 -3.251e-078	1.307 7e-077 1.683 5e-077	8.244 8e-079 -2.062 6e-076
	2*MaxGen*/4	2.198 9e-106	2.846 1e-112	1.212 9e-057 -1.913 1e-057	4.085 9e-058 -7.793 9e-058	2.130 2e-057 -1.041 5e-056	3.716 5e-058 1.338e-057	1.110 3e-057 8.399 8e-058
	3*MaxGen*/4	5.288 8e-060	1.539 9e-064	2.879e-033 -5.489 6e-035	-3.357 4e-036 -2.829 6e-034	3.038 6e-033 1.054 5e-036	1.750 5e-033 2.787 1e-033	-1.378 3e-033 3.677 4e-033

注：上表中的“——”表示该算法中不设定代数阈值。

表 2　对 Rastrigrin 函数的优化结果

优化方法	代数阈值	平均收敛值	最优收敛值	最优收敛值对应变量				
Cω-PSO	——	1.894 7e+001	1.220 6e+001	0.009 781 6 -0.008 669 8	-0.995 71 -0.974 32	1.975 2 0.001 240 5	0.994 37 0.015 83	-1.976 1 1.004 5
LDω-PSO	——	2.908 3e+001	1.069 1e+001	-0.976 7 1.017	1.017 2 -1.962	-0.986 04 0.984 55	0.029 66 1.01	-0.013 055 -0.013 671
Cω-GMPSO	*MaxGen*/4	1.049 3e-006	0	-9.129 1e-015 -7.547 9e-016	-2.646 1e-013 -4.090 5e-014	-4.435e-017 9.964 2e-010	7.901 7e-014 7.209 8e-014	6.294e-017 2.401 9e-017
	2*MaxGen*/4	2.793e-010	0	4.998e-011 7.757 8e-012	-3.081 4e-010 2.891 1e-011	-6.385 8e-010 -3.893 4e-010	-4.579 7e-012 5.403 6e-011	1.070 9e-010 -2.277e-010
	3*MaxGen*/4	5.674 9e-001	3.1810e-006	3.503e-008 1.714 9e-007	9.227 8e-008 -5.396 6e-008	-8.692 3e-010 3.355 1e-008	3.796 4e-010 1.722 3e-007	1.546 7e-005 -2.911 2e-008
LDω-GMPSO	*MaxGen*/4	0	0	-4.139 6e-014 -8.253 3e-015	-5.317 8e-011 2.892 8e-016	-1.020 9e-015 8.911 4e-010	1.630 7e-013 6.310 2e-015	-6.442 5e-015 1.079e-012
	2*MaxGen*/4	0	0	-3.991 9e-014 6.990 4e-013	-1.15e-015 5.457 8e-012	-6.123 6e-010 -4.555 3e-012	-7.465 5e-012 5.945 4e-011	-1.522 8e-015 -7.273 1e-012
	3*MaxGen*/4	0	0	1.544 8e-010 -1.856 3e-011	7.554 2e-014 8.484 6e-013	2.147 3e-010 -1.128 5e-014	-4.934 6e-012 7.561 9e-011	2.212 9e-012 3.982 9e-010

注：上表中的“——”表示该算法中不设定代数阈值。



（a）代数阈值为 *MaxGen*/4　（b）代数阈值为 2*MaxGen*/4　（c）代数阈值为 3*MaxGen*/4

图 1　Sphere 函数在优化方法 LDω-GMPSO 下的最大聚集距离（MaxDist）变化曲线

GMPSO 下的最大聚集距离（MaxDist）变化曲线）。

从图 1 和图 2 中可以看出：MaxDist 曲线随着迭代的进行均呈现逐步下降趋势，反映了粒子在搜索空间的“趋同”过程。论文通过增加代数阈值和基于最大聚集距离的高斯变异策略，使得这种“趋同”出现了相对平缓的下降，也就是说，在维持粒子群多样性的同时，对搜索空间进行进一步的“探测”，从而有效防止了早熟收敛现象的发生。

为了进一步检验文中所提出 PSO 算法的优化性能，论文将其对 Sphere 和 Rastrigrin 函数的优化结果与文献[10-13]进行了比较（如表 3 所示）。

从表 3 易知：论文中算法 LDω-GMPSO 与 COPSO 和 HPSO 取得了同样的优化结果；LDω-GMPSO 比 PSO-MO 和 RPPSO（除 Sphere 函数优化结果外）取得了更优的测试结果。这充分说明本文算法的性能均达到或者优于文献中的已有算法，也进一步说明了本文算法的可行性和有效性，在进行高维、多峰的纯数值函数求解时能取得较为满意的优化结果。

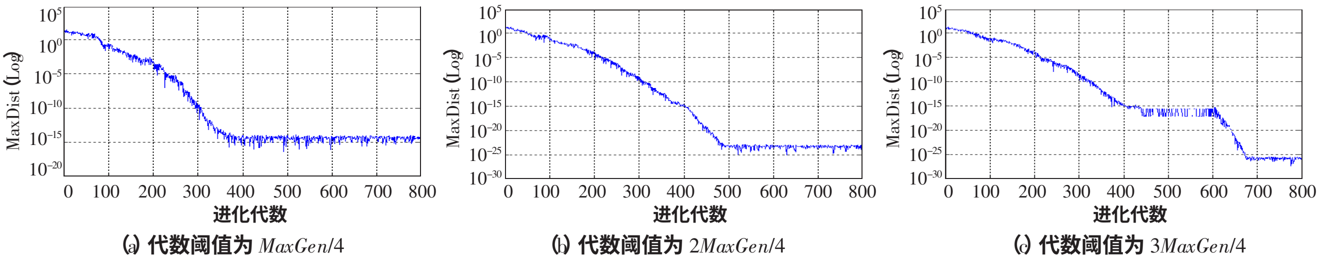


图 2　Rastrigrin 函数在优化方法LDω–GMPSO下的最大聚集距离(MaxDist)变化曲线

表 3　本文算法与文献[10–13]中算法优化结果比较

测试函数	优化算法	最优收敛值	平均收敛值
Sphere	RP–PSO[12]	0	0
	COPSO[10]	0	0
	HPSO[13]	0	0
	PSO–MO[11]	8.20e–016	4.91e–011
	LDω–GMPSO	3.82e–151	3.88e–143
Rastrigrin	RP–PSO[12]	2.984 8	19.646 9
	COPSO[10]	0	0
	HPSO[13]	0	0
	PSO–MO[11]	6.73e–016	8.48e–011
	LDω–GMPSO	0	0

5　结论

粒子群优化算法是一种基于群体智能的随机全局优化进化算法。论文针对 PSO 算法在后期易陷入局部最优的缺陷，在深入分析和研究基本粒子群优化算法的前提下，提出了一种基于进化代数阈值和粒子间最大聚集距离高斯变异的粒子群优化算法，以使基本 PSO 算法在后期摆脱局部极值点的束缚，提高收敛速度和搜索精度。测试函数仿真结果表明了本文算法的可行性和有效性。

参考文献：

[1] Kennedy J，Eberhart R C.Particle swarm optimization[C]//Proc IEEE International Conference on Neural Networks.IV Piscataway，NJ：IEEE Service Center，1995：1942–1948.

[2] Kennedy J，Mendes R.Population structure and particle swarm performance[C]//Proceedings of the IEEE Congress on Evolutionary Computation，2002：1671–1676.

[3] Higasshi N.Hitoshi particle swarm optimization with Gaussian mutation[C]//Proc of the Congress on Evolutionary Computation，2003：72–79.

[4] Ratnaweera A，Halgamuge S K.Self–organizing hierarchical particle swarm optimizer with time–varying acceleration coefficients[J].IEEE Transactions on Evolutionary Computation，2004，8(3)：240–255.

[5] Monson C K，Sepp I K D.The kalman swarm–a new approach to particle motion in swarm optimization[C]//Proceedings of the Genetic and Evolutionary Computation Conference.[S.l.]：Springer，2004：140–150.

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

[7] Rodriguez A，Reggia J A.Extending self–organizing particle systems to problem solving[J].Artificial Life，2004，10(4)：379–395.

[8] Katare S，Kalos A，West D.A hybrid swarm optimizer for efficient parameter estimation[C]//Proceedings of the IEEE Congress on Evolutionary Computation，2004：309–315.

[9] Sun J，Feng B.Particle swarm optimization with particles having quantum behavior[C]//Proceedings of the IEEE Congress on Evolutionary Computation，2004：325–331.

[10] Liu H Y，Lin Y E.A hybrid particle swarm optimization based on chaos strategy to handle local convergence [J].Computer Engineering and Applications，2006，42(13)：77–79.

[11] Zhao Z G，Su Y D.Particle swarm optimization with mutation operator[J].Computer Engineering and Applications，2006，42(13)：45–47.

[12] Yu B H，Yuan X H.A random perturbation particle swarm optimization algorithm[J].Computer Engineering，2006，32(12)：189–190.

[13] Li L，Li H Q.Solving for complex functions with high dimensions based on hybrid particle swarm optimization[J].Computer Applications，2007，27(7)：1754–1756.

[14] Shi Y，Eberhart R C.Particle swarm optimization：developments，applications and resources[C]//Proc Congress on Evolutionary Computation 2001.NJ，Piscataway：IEEE Press，2001：81–86.

[15] Shi Y，Eberhart R C.A modified particle swarm optimizer[C]//IEEE World Congress on Computational Intelligence，1998：69–73.

[16] Maurice C，Kennedy J.The particle swarm–explosion，stability and convergence in a multidimensional complex space[J].IEEE Transaction on Evolutionary Computation，2002，6(1)：58–73.

[17] Lu Z S，Hou Z R.Particle Swarm Optimization with adaptive mutation[J].Acta Electronica Sinica，2004，32(3)：416–420.

[18] van den Bergh F.An analysis of particle swarm optimizers[D].Department of Computer Science，University of Pretoria，South Africa，2002：81–83.

[19] Zhang L P，Yu H J.Analysis and improvement of particle swarm optimization algorithm[J].Information and Control，2004，33(5)：513–517.

[20] Li N，Liu F，Sun D B.A study on the particle swarm optimization with mutation operator constrained layout optimization[J].Chinese Journal of Computers，2004，27(7)：897–903.