# 一种带混沌变异的粒子群优化算法

朱红求 阳春华 桂卫华 李勇刚
(中南大学信息科学与工程学院 长沙 410083)

**摘 要** 为了克服粒子群算法在进化后期存在收敛速度慢、易陷入局部极小等问题,提出了一种混沌变异粒子群优化算法。该算法根据群体适应度变化率对种群中非优胜粒子进行变异操作,并对全局最优位置进行小范围混沌扰动,以增强算法跳出局部最优的能力。对几种复杂典型函数与标准粒子群算法进行了仿真测试,结果表明该算法明显改善了全局搜索能力和抗早熟收敛性能。

**关键词** 粒子群,混沌变异,早熟收敛
**中图法分类号** TP301 **文献标识码** A

## Particle Swarm Optimization with Chaotic Mutation

ZHU Hong-qiu YANG Chun-hua GUI Wei-hua LI Yong-gang
(School of Information Science&Engineering, Central South University, Changsha 410083, China)

**Abstract** To overcome the disadvantage of low convergence speed and the premature convergence during the later computation period of particle swarm optimization, a chaotic particle swarm optimization (CPSO) was proposed. Aimed to improve the ability to break away from the local optimum and to find the global optimum, the non-winner particles were mutated by chaotic search and the global best position was mutated using the small extent of disturbance according to the variance ratio of population's fitness. The numerical simulation comparing to the standard PSO was performed using of complex benchmark functions with high dimension. The results show that the proposed algorithm can effectively improve both the global searching ability and much better ability of avoiding pre-maturity.

**Keywords** Particle swarm optimization, Chaotic mutation, Premature convergence

## 1 引言

粒子群优化算法(PSO)作为一种基于种群搜索策略的全局优化进化算法[1],具有概念简单、容易实现等特点,在进化初期收敛速度快,需要调整的参数少。PSO成为当前优化算法研究领域的一大热点,已成功应用于函数优化、图像处理、负荷预测、工业过程控制等领域[2-6]。

与其他进化算法类似,PSO在实际应用中也存在过早收敛等问题,特别是在进化后期容易陷入局部极小点,收敛速度慢[7,8],所能达到的精度较差。本文针对标准PSO存在的早熟与收敛速度较慢等问题,利用混沌序列的随机性、遍历性及规律性等特点,提出了一种基于混沌序列的改进型粒子群算法。该算法根据群体适应度变化率的情况,在种群进入局部收敛后对性能较差的粒子以及全局最优值进行混沌变异,以改善PSO跳出局部最优点的能力。采用测试函数对标准PSO和提出的混沌PSO算法进行了比较研究,结果表明,所提出的改进算法明显提高了粒子群算法的全局收敛性能,并能有效避免粒子群优化算法中的早熟收敛问题。

## 2 粒子群优化算法

PSO是基于群体与适应度的。粒子群的个体(粒子)代表问题的一个可能解,每个粒子具有位置和速度两个特征,粒子位置坐标对应的目标函数值可作为该粒子的适应度,算法通过适应度来衡量粒子的优劣。算法首先初始化一群随机粒子,然后通过迭代找到最优解。速度与位置更新公式为:

$$v_{id}^{t+1}=w\cdot v_{id}^{t}+c_1\cdot rand_1()\cdot(g_{best}-x_{id}^{t})+c_2\cdot rand_2()\cdot(p_{best}-x_{id}^{t}) \quad (1)$$

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

其中,$v_{id}^{t}$是粒子的速度;$x_{id}^{t}$是粒子的当前位置向量,$p_{best}$与$g_{best}$为某一粒子和整个粒子群迄今为止搜索到的最优位置。$rand_{id}$,$rand_{gd}$为[0,1]之间的随机数,$c_1$,$c_2$为学习因子,通常,$c_1=c_2=2$。$w$为惯性权值,一般在0.1到0.9之间取值,文献[2,7]通过大量实验证明,如果$w$随算法迭代的进行而线性减小,将显著改善算法的收敛性能。设$w_{max}$,$w_{min}$分别是$w$的最大值和最小值,$iter$,$iter_{max}$分别是当前迭代次数和最大迭代次数。则有:

到稿日期:2009-04-24 返修日期:2009-07-20 本文受国家自然科学基金(60874069,60804037),湖南省自然科学基金(09JJ3122)和国家863项目(2009AA04Z124,2009AA04Z137)资助。

朱红求 男,博士生,研究领域为复杂过程信息处理与优化建模;阳春华 女,博士生导师,研究领域为复杂工业过程优化控制、智能自动控制系统等。

$$w=w_{\max}-iter\times\frac{w_{\max}-w_{\min}}{iter_{\max}} \tag{3}$$

更新过程中，粒子每一维的最大速率限制在 $v_{max}$，粒子每一维的坐标也被限制在允许范围内。同时，$p_{best}$ 与 $g_{best}$ 在迭代过程中不断更新，最后输出的 $g_{best}$ 就是算法得到的最优解。

从式(1)可以看出，粒子在下一时刻的速度是由当前时刻的速度、当前位置与该粒子的局部最优位置的距离、当前位置与全局最优位置的距离共同决定的。在粒子群算法后期，如果粒子群的最优个体位置以及全局最优位置变化很小，其速度更新将主要由 $w\cdot v_{id}$ 来决定。而从权值计算式(3)可以看出，迭代次数越大，$w$ 越小，不断迭代导致速度也越来越小，这意味着粒子群无法在解空间内重新搜索而陷入局部最优，即出现了停滞或早熟收敛现象。

为了克服这一缺陷，当群体的最优适应值长时间未发生变化时，应增加扰动而使粒子群跳出局部最优位置。定义最优适应值变化率 $k$：

$$k=\frac{|f(t)-f(t-T)|}{|f(t-T)|} \tag{4}$$

其中，$f(t)$ 为种群第 $t$ 代的最优适应值，$f(t-T)$ 为种群第 $(t-T)$ 代的最优适应值，则 $k$ 表示种群在最近进化 $T$ 代内最优适应值的相对变化率。当 $k$ 小于某一设定阈值时，则认为粒子群算法进入了局部最优点，从而启动混沌变异算法，通过混沌扰动协助粒子群算法跳出局部最优点。

## 3 一种带混沌变异的粒子群优化算法

混沌运动是存在于非线性系统中的一种较为普遍的现象[9,10]，能在一定范围内按其自身规律不重复地遍历所有状态。因而混沌成为优化搜索且能避免陷入局部极小的一种新颖的优化技术，具有全局性的优点。然而通常情况下，该方法产生的随机数不可能完成连续变量空间的遍历搜索。

混沌优化算法的基本思想就是把混沌变量线性映射到优化变量的取值区间，然后利用混沌变量进行搜索。混沌动力学模型种类繁多，这里选用一种最为简单的一维非线性映射模型，即描述生物种群演变规律的简化模型 Logistic 映射：

$$\begin{aligned}&x(t+1)=\mu x(t)(1-x(t))\\&t=1,2,\cdots,x(0)\in[0,1]\end{aligned} \tag{5}$$

其中，$\mu$ 为控制参量，当 $\mu=4$ 时，Logistic 映射是[0,1]区间上的满映射，且系统处于完全的混沌状态，我们称式(5) 产生的序列$\{x(t)\}$为混沌变量。在优化过程中，利用该映射产生的混沌变量进行搜索，搜索过程按混沌运动自身的规律进行，其随机性保证了大范围搜索能力，遍历性使算法按系统自身的规律不重复地遍历所有可能的状态，有利于克服一般随机算法中以分布遍历性搜索带来的局限性。

基于混沌优化的思想，在种群进化过程中，根据最优适应度变化率来判断是否早熟，当变化率小于设定的值时，选择粒子群中的优胜粒子按照式(1)、式(2)进行速度和位置的更新，而非优胜粒子则映射为混沌变量，进行混沌优化后，与更新后的优胜粒子组成新的种群；同时对当前的全局最优值也采用混沌优化方法进行小范围的扰动，通过混沌优化操作使种群跳出局部最优点，增大寻找全局最优解的几率。此外，在种群初始化时，采用混沌变量对粒子的位置和速度进行初始化，这样可以在保证初始变量随机性的前提下，提高种群的多样性和遍历性。归纳起来，混沌粒子群算法的基本步骤如下：

Step1 初始化。确定群体规模 $N$，进化总代数 $M$。选用式(5)所示的 Logistic 映射，给式(5)赋 $n$ 个微小差异的初值，得到 $n$ 个混沌变量，并将混沌变量映射到位置向量的取值区间，随机选择其中的 $N$ 个粒子构成初始种群；

Step2 评价。计算种群中个体的适值，并按适值高低进行排序，计算局部最优值和全局最优值。如个体中的目标值有满足终止条件的，则结束，否则，进入下一步；

Step3 速度和位置的更新。将适值较高的 85% 的优胜个体按照式(1)、式(2)进行速度和位置的更新，得到子种群 Popul 1；种群中适值较低的 15% 的非优胜个体 $x^{(k)}$ 按照式(6)映射为混沌变量：

$$c_x^{(k)}=\frac{x^{(k)}-x_{\min}}{x_{\max}-x_{\min}} \tag{6}$$

其中，$x_{\max}$ 和 $x_{\min}$ 分别表示 $x^{(k)}$ 位置向量的上界和下界。将得到的混沌变量按照式(5)进行变异后得到 $x^{(k+1)}$，并根据式(7)映射为位置向量：

$$x^{(k+1)}=x_{\min}+cx^{(k+1)}(x_{\max}-x_{\min}) \tag{7}$$

这样就得到变异后的子种群 Popul 2。将 Popul 1 和 Popul 2 组成新的种群。

Step4 对全局最优值进行小范围的混沌变异。设定混沌变异运行的次数，并将 $g_{best}$ 映射到区间[0,1]后形成变量 $\xi$，则全局最优值的混沌变异按照式(8)进行：

$$\xi^*=(1-a)\xi+a\sigma_k \tag{8}$$

其中，$\xi^*$ 为混沌变异后的位置向量，$\sigma_k$ 为迭代 $k$ 次后的混沌变量。反复迭代直到达到运行次数最大值或者得到更优的全局最优位置向量。对全局最优位置向量仅进行小范围的扰动，以防止混沌变量接近全局最优解，此时 Logistic 映射产生的切比雪夫型分布的混沌序列会影响搜索的效率和能力，特别是维数较大的情况下，效果特别明显。

Step5 检查是否满足结束条件，若满足则结束寻优；否则返回 Step2 继续迭代。

在上述算法中，对一些适应值最差的非优胜个体采用混沌变异操作，而对大部分适应值较高的优胜个体进行粒子群算法的基本操作，以保证在优化算法过程中，优秀个体不会受到“破坏”，从而保持了粒子群算法的优点；同时对全局最优向量进行小范围的扰动，通过这两种变异方法，帮助算法跳出局部极小，从而改善算法的性能。但是由于在变异过程中增加了一些变异运算环节，因此不可避免地会增加算法的运算时间。

## 4 仿真比较分析

下面将通过 3 个典型函数优化问题(求解最小值)来测试本文算法的性能，同时与粒子群优化算法进行比较。Rosenbrock 函数是很难极小化的病态单峰二次函数；Rastrigin 函数和 Griewank 函数为多峰函数。

(1)Rosenbrock 函数：

$$\begin{aligned}&f_1(x)=\sum_{i=1}^{n}(100\times(x_{i+1}-x_i^2)^2+(x_i-1)^2),\\&-30\leqslant x_i\leqslant 30\end{aligned} \tag{9}$$

其最优状态和最优值为 $\min f(x^*)=f(1,1,\cdots 1)=0$。

(2)Rastrigin 函数：

$$f_2(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10),-5.12\leqslant x_i\leqslant 5.12 \tag{10}$$

其最优状态和最优值为 $\min f(x^*)=f(0,0,\cdots,0)=0$。

(3)Griewank 函数：

$$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left|\frac{x_i}{\sqrt{i}}\right|+1,-600\leqslant x_i\leqslant 600 \tag{11}$$

其最优状态和最优值为 $\min f(x^*)=f(0,0,\cdots 0)=0$。

为了评价不同算法的优劣，采用一组评估指标：平均最优适应值 $F_{ave}$、平均成功率 $R_{as}$、平均运算时间 $T_{ave}$，分别定义如下：

① 平均最优适应值 $F_{ave}$：$M$ 次试验所得函数值的算术平均值。

② 平均成功率 $R_{as}$：进行 $M$ 次试验，其中 $Ms$ 次寻优成功，则平均成功率：

$$R_{as}=(Ms/M)*100\% \tag{12}$$

③ 平均运算时间 $T_{ave}$：进行 $M$ 次试验，以 $T_n$ 表示第 $n$ 次寻优所需时间，则平均运算时间(s)：

$$T_{ave}=\sum_{n=1}^{M}T_n/M \tag{13}$$

仿真分析实验采用 Matlab 7.0 以及 Pentium IV 2.4 GHz，512M RAM 的计算机，实验在 10 维空间中进行。对于所有实验，算法的最大迭代次数都设定为 1000 次，种群的大小设为 50，最大速度为测试函数的取值范围。惯性因子 $w$ 为 0.9～0.4，学习因子 $c_1$ 和 $c_2$ 的取值设为 2.0。$f_1(x)$，$f_2(x)$ 和 $f_3(x)$ 的收敛精度依次取 10，5 和 0。为了消除偶然因素，对每个函数的寻优实验都进行 30 次，同时在对各个函数测试时对其单次试验的最优值进行限幅，以防止个别比较差的结果影响测试计算结果。上述 3 个函数的试验结果如表 1 所列。

表 1　两种算法的测试结果比较

| function | method | $f_{ave}$ | $R_{as}$ | $T_{ave}$ |
|---|---|---|---|---|
| $f_1$ | PSO | 10.9244 | 73.3 | 1.2232 |
| | CPSO | 4.0523 | 96.7 | 9.0785 |
| $f_2$ | PSO | 6.5336 | 60 | 1.367 |
| | CPSO | 3.2558 | 93.3 | 22.458 |
| $f_3$ | PSO | 0.1427 | 76.7 | 1.313 |
| | CPSO | 0.1031 | 83.3 | 21.658 |

从表 1 中可以看出，所测试的 3 个函数混沌粒子群算法总体上能获得比基本粒子群算法更优的结果。对于 Rosenbrock 函数，CPSO 的寻优结果与函数本身的全局极值还存在一定的差距，但是相对于 PSO 算法的寻优结果，还是有一定的优势。而在对两个多峰函数 Rastrigin 和 Griewank 的实验中，CPSO 对 Rastrigin 函数的寻优有明显的改进，而对 Griewank 函数的寻优改进较小。此外，最大运行时间不超过 30 秒，对于一般的工业过程优化来说，能满足过程的实时性要求。

CPSO 和 PSO 的寻优过程比较如图 1 所示。从图 1 中可以看出，CPSO 的初始值好于 PSO 的初始值，这是由于在 CPSO 算法中采用混沌序列初始化，保证了初始值分布的全局性，而 PSO 采用随机初始化，较 CPSO 而言难以找到比较合理的初始值。同时还可以看出，混沌扰动的力度和寻优时间存在矛盾，在算法后期，由于小范围的扰动对跳出局部极小效果不是很明显，因此影响了混沌粒子群算法的性能。

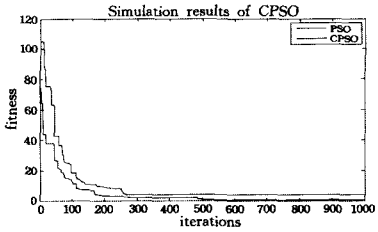


图 1　Rastrigin 函数寻优过程对比

**结束语**　本文针对粒子群优化算法的早熟收敛问题，提出了一种采用混沌变异机制的粒子群优化算法。该算法根据群体适应度变化率作为早熟判断机制，对早熟条件下的非优胜粒子采用混沌搜索方法进行变异，同时对全局最优值进行小范围内扰动，以增强粒子群算法跳出局部最小的能力和收敛性能。仿真试验表明，混沌粒子群算法有效提高了全局搜索能力和抗早熟性能，可以用于过程优化控制。考虑到变异率和扰动幅度会影响算法的性能和计算时间，因此这两个参数可以根据实际情况进一步优化。

## 参 考 文 献


[1] Kennedy J, Eberhart R. Particle Swarm Optimization[C]// Proc. IEEE Int. Conf. on Neural Networks. Perth, 1995: 1942-1948

[2] Li L Y, Li D R. Fuzzy entropy image segmentation based on particle swarm optimization [J]. Progress in Natural Science, 2008, 18(9): 1167-1171

[3] Siahkali H, Vakilian M. Electricity generation scheduling with large-scale wind farms using particle swarm optimization[J]. Electric Power Systems Research, 2009, 79(5): 826-836

[4] Baskar G, Mohan M R. Contingency constrained economic load dispatch using improved particle swarm optimization for security enhancement[J]. Electric Power Systems Research, 2009, 79(4): 615-621

[5] Samanta B, Nataraj C. Use of particle swarm optimization for machinery fault detection[J]. Engineering Applications of Artificial Intelligence, 2009, 22(2): 308-316

[6] Gaing Z L. A Particle Swarm Optimization Approach for Optimum Design of PID Controller in AVR System[J]. IEEE transactions on energy conversion, 2004, 19(2): 384-391

[7] Zhang C S, Sun J G. An alternate two phases particle swarm optimization algorithm for flow shop scheduling problem [J]. Expert Systems with Applications, 2009, 36(3): 5162-5167

[8] Jiang Y, Hu T S, Huang C C, et al. An improved particle swarm optimization algorithm[J]. Applied Mathematics and Computation, 2007, 193(1): 231-239

[9] LU H J. A new optimization algorithm based on Chaos[J]. J of Zhejiang Univ SCIENCE A, 2006, 7(4): 539-542

[10] Tavazoei M S, Haeri M. An optimization algorithm based on chaotic behavior and fractal nature[J]. Journal of Computational and Applied Mathematics, 2007, 206(2): 1070-1081