文章编号：1001-0920(2009)10-1545-04

# 基于混沌的弹性粒子群全局优化算法

高雷阜，刘旭旺
（辽宁工程技术大学 数学与系统科学研究所，辽宁 阜新 123000）

**摘 要**：为了克服粒子群优化容易陷入局部极小的缺陷，利用粒子速度不依赖于其与最优粒子之间距离的大小，而仅依赖其方向信息的特点，采用自适应策略弹性地修正粒子速度的幅值. 同时，充分利用混沌运动的遍历性、随机性及对初值的敏感性等特性，提出一种基于混沌的弹性粒子群优化(CRPSO)算法，并将其成功用于典型多极点函数优化. 仿真结果表明，该算法增强了摆脱局部极值点的能力，提高了收敛速度和精度.

**关键词**：非线性规划；全局优化；粒子群；混沌优化；弹性修正
**中图分类号**：TP18 **文献标识码**：A

# Resilient particle swarm global optimization algorithm based on chaos

*GAO Lei-fu*, *LIU Xu-wang*
(Institute of Mathematics and Systems Science, Liaoning Technical University, Fuxin 123000, China. Correspondent: LIU Xu-wang, E-mail: liuxuwang007@163.com)

**Abstract**: To overcome the vice that the particle swarm optimization is prone to trap into local minima, by using a strategy in which the velocity is not dependent on the size of distance between the individual and the optimal particle but only dependent on its direction, an adaptive scheme is adopted to adjust the magnitude of the velocity resiliently. At the same time, by making the best of the ergodicity, stochastic property and regularity of chaos, a resilient particle swarm global optimization algorithm based on chaos is proposed, which is applied to optimize the functions having many apices. Simulation results show that the new algorithm has the ability to avoid being trapped in local minima, and improves computational precision and convergence speed.

**Key words**: Nonlinear optimization; Global optimization; Particle swarm; Chaos optimization; Resilient adjustment

## 1 引 言

粒子群优化（PSO）算法是 Kennedy 和 Eberhart[1]于 1995 年提出的一种智能优化算法. PSO 算法源于对鸟群和鱼群等生物群体运动行为的研究，和蚁群算法一样都有群智能的特点[2]. 由于其简单，收敛速度快，且对目标函数要求较少等特点，发展十分迅速，在诸多领域得到成功应用. 与其他智能算法类似，PSO 算法也存在早熟收敛和局部寻优能力差等缺点. 目前的解决方法主要是增加种群的多样性以及和其他方法融合等[3-4].

造成 PSO 算法早熟收敛的主要原因[5]是：在算法后期，所有粒子将聚集到极值点附近，各粒子的速度将趋于零. 如果这是一个局部极值点，则所有粒子将很难跳出其约束范围，从而导致算法早熟收敛. 对此，本文将混沌和弹性粒子群有机结合，充分利用混沌的动力学特征，使混沌序列更具均匀性和遍历性. 计算粒子的速度时，不考虑它与最优粒子之间距离的大小，而只利用其方向信息，并采用一种自适应策略弹性地修正粒子速度的幅值. 同时，利用早熟判断机制来调整惯性权重，提出一种基于混沌的弹性粒子群优化(CRPSO)算法，给出了算法流程，并将其成功用于多极点函数优化.

## 2 一般混沌粒子群全局优化算法

### 2.1 基本粒子群算法

在 PSO 模型中，每个粒子有一个速度决定其飞翔的方向和距离. PSO 算法初始化为一群随机粒子，然后粒子开始追随当前的最优粒子运动，直到在整个解空间中搜索到最优解为止. 在每次迭代中，粒

**收稿日期**：2008-12-11；**修回日期**：2009-03-12.
**基金项目**：辽宁省自然科学基金项目(20042176).
**作者简介**：高雷阜(1963—)，男，辽宁阜新人，教授，博士生导师，从事非线性优化理论与应用、非线性动力系统预测的研究；刘旭旺(1983—)，男，河南漯河人，硕士生，从事最优化理论与方法的研究.

子通过追踪两个极值来更新自己. 一个是个体极值 pbest;另一个是全局极值 gbest.

假设在 $D$ 维搜索空间中,有 $m$ 个粒子组成一群体,第 $i$ 个粒子的位置为 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$,第 $i$ 个粒子经历过的最好位置记为 $P_i$,每个粒子的飞行速度为 $V_i=(v_{i1},v_{i2},\cdots,v_{iD})$, $i=1,2,\cdots,m$.

整个群体中,所有粒子经历过的最好位置为 $P_g$,每代粒子按下式更新自己的速度和位置:

$$v_{id}(t+1)=wv_{id}(t)+c_1r_1(p_{id}-x_{id}(t))+c_2r_2(p_{gd}-x_{id}(t)),\tag{1}$$

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

其中:$w$ 为惯性权重;$c_1$,$c_2$ 为学习因子;$r_1$,$r_2$ 为[0,1]之间的随机数.

PSO 算法的详细介绍和算法流程见文献[5].

### 2.2 混沌粒子群算法基本思想

混沌粒子群优化(CPSO)算法的基本思想主要体现在以下两个方面:

1) 采用混沌序列初始化粒子的位置和速度,既不改变初始化时所具有的随机性本质,又提高了种群的多样性和粒子搜索的遍历性,在产生大量初始群体的基础上,从中择优出初始群体.

2) 以当前整个粒子群迄今为止搜索到的最优位置为基础产生混沌序列,用序列中的最优粒子位置替代当前粒子群中的一个粒子的位置,可在迭代中产生局部最优解的许多邻域点,以此帮助"惰性"粒子逃离局部极小点,并快速搜寻到最优解.

## 3 基于混沌的弹性粒子群全局优化算法

假设要解决的优化问题为

$$\min f(x_1,x_2,\cdots,x_n),\ \text{s.t. } a_i\leqslant x_i\leqslant b_i.\tag{3}$$

采用逻辑自映射函数产生混沌,其模型为

$$z_{n+1}=1-2\times z_n^2,\quad n=1,2,\cdots,\ 1<z_n<1.\tag{4}$$

在 PSO 算法中,$w$ 对算法能否收敛具有重要作用. 由于 PSO 算法的实际搜索过程是非线性的且是高度复杂的,致使 $w$ 线性递减的策略不能反映实际的优化搜索过程,使算法在求解复杂问题的后期易发生早熟收敛. 对此,本文借助早熟收敛程度和个体适应值来调整 $w$,同时将混沌和弹性改变粒子速度有机结合,提出一种基于混沌的弹性粒子群全局优化算法.

### 3.1 早熟收敛程度评价

全局最优值总是优于所有个体当前的适应度值. 设粒子群的规模数是 $N$,则有

$$f_{\text{avg}}=\frac{1}{N}\sum_{i=1}^{N}f_i,\tag{5}$$

其中 $f_i$ 为粒子当前适应值. 设最优粒子的适应值为 $f_g$,并将优于 $f_{\text{avg}}$ 的适应值求平均得到 $f'_{\text{avg}}$. 定义 $\Delta=|f_{\text{avg}}-f'_{\text{avg}}|$,$\Delta$ 可用来评价粒子群的早熟收敛程度,$\Delta$ 越小说明粒子群越趋于早熟收敛.

### 3.2 自适应调整策略

适应值为 $f_i$ 的粒子,$w$ 的具体调整方法如下:

$$w=\begin{cases}w-(w-w_{\min})\left|\dfrac{f_i-f'_{\text{avg}}}{f_g-f'_{\text{avg}}}\right|, & f_i\text{ 优于 }f'_{\text{avg}};\\ w_{\min}+(w_{\max}-w_{\min})\times\dfrac{1+\cos((\text{iter}-1)\pi/(\max\text{step}-1))}{2}, & f_i\text{ 优于 }f_{\text{avg}}\text{ 但次于 }f'_{\text{avg}};\\ 1.5-\dfrac{1}{1+k_1\exp(-k_2\Delta)}, & f_i\text{ 次于 }f_{\text{avg}}.\end{cases}\tag{6}$$

其中:$w_{\min}$ 为 $w$ 的最小值,$w_{\max}$ 为搜索开始时的最大值,iter 为当前迭代步数,maxstep 为允许最大迭代步数. $k_1$ 的选取应使式(6)能提供大于 1 的惯性权重,$k_2$ 主要用来控制式(6)的调节能力,若 $k_2$ 过大,则会加快收敛,使算法在早期全局寻优能力不足;若 $k_2$ 过小,则在后期算法不能有效地跳出局部最优.

### 3.3 弹性处理粒子速度

在算法后期,所有粒子都聚集到一个极值点附近[6],此时,$p_{id}$,$p_{gd}$,$x_{id}$ 相差很小,因此粒子的飞行速度 $v_{id}$ 将趋近于 0. 如果该极值是一个局部极值点,则粒子很难跳出其约束范围,直至所有粒子都集中到该局部极值点上,因而引起算法的早熟收敛.

为了改善算法的全局搜索能力,必须避免算法后期粒子的飞行速度过小. 在本文的算法中,粒子飞行的方向与原算法一样,但飞行速度的大小不同,它不是由该粒子与最优粒子之间距离的大小决定,而是一种自适应弹性的修正,即

$$v_{id}(t+1)=wv_{id}(t)+c_1r_1\Delta v_{id}^1(t)+c_2r_2\Delta v_{id}^2(t),\tag{7}$$

其中

$$\begin{cases}\Delta v_{id}^1(t)=\Delta_{id}^1(t)\text{sgn}[p_{id}(t)-x_{id}(t)],\\ \Delta v_{id}^2(t)=\Delta_{id}^2(t)\text{sgn}[p_{gd}(t)-x_{id}(t)].\end{cases}\tag{8}$$

这里:sgn(·)为符号函数,$\Delta_{id}^1$ 和 $\Delta_{id}^2(t)$ 采用如下策略弹性修正. 当粒子前后两次飞行方向一致时,说明粒子正从极值(个体或全局极值)的一侧向其逼近,可加大 $\Delta_{id}^1(t)$(或 $\Delta_{id}^2(t)$),以加快收敛速度;如果前后两次飞行方向不一致,则说明该粒子正在极值附近徘徊,此时应减小 $\Delta_{id}^1(t)$(或 $\Delta_{id}^2(t)$),以避免在极值点附近徘徊过久;如果粒子前后两次迭代有一次与极值重合,则不改变 $\Delta_{id}^1(t)$(或 $\Delta_{id}^2(t)$). 于是,可以按下式修正 $\Delta_{id}^1(t)$ 和 $\Delta_{id}^2(t)$:

$$\Delta_{id}^{1}(t)=\begin{cases}\theta_{+}\,\Delta_{id}^{1}(t-1),\\ \quad [p_{id}(t)-x_{id}(t)][p_{id}(t-1)-\\ \quad x_{id}(t-1)]>0;\\ \theta_{-}\,\Delta_{id}^{1}(t-1),\\ \quad [p_{id}(t)-x_{id}(t)][p_{id}(t-1)-\\ \quad x_{id}(t-1)]<0;\\ \Delta_{id}^{1}(t-1),\ \text{others};\end{cases}\tag{9}$$

$$\Delta_{id}^{2}(t)=\begin{cases}\theta_{+}\,\Delta_{id}^{2}(t-1),\\ \quad [p_{gd}(t)-x_{gd}(t)][p_{gd}(t-1)-\\ \quad x_{gd}(t-1)]>0;\\ \theta_{-}\,\Delta_{id}^{2}(t-1),\\ \quad [p_{gd}(t)-x_{gd}(t)][p_{gd}(t-1)-\\ \quad x_{gd}(t-1)]<0;\\ \Delta_{id}^{2}(t-1),\ \text{others}.\end{cases}\tag{10}$$

其中：$\theta_+$ 和 $\theta_-$ 分别为递增、递减因子，满足 $0<\theta_-<1<\theta_+$；同时，为了保证算法的快速收敛性，$\Delta_{id}^{1}(t)$ 和 $\Delta_{id}^{2}(t)$ 必须限定在一个合理的范围内，即 $\Delta_{id}^{1}(t)$，$\Delta_{id}^{2}(t)\in[\Delta_{\min},\Delta_{\max}]$.

### 3.4　基于混沌的弹性粒子群全局优化算法流程

该算法是在 CPSO 算法的基础上，根据各个粒子前后两次飞行方向来调整速度大小的一种改进混沌粒子群优化算法. 算法具体流程如下：

Step1：初始化设置最大允许迭代次数或适应度误差限，以及相关参数：惯性权重、学习因子等.

Step2：混沌初始化粒子位置和速度.

1）随机产生一个 $n$ 维向量，$z_1=(z_{11},z_{12},\cdots,z_{1n})$，$z_{ij}\in[-1,1]$，$n$ 为目标函数的变量个数，由式(4)可得 $N$ 个向量 $z_1,z_2,\cdots,z_N$；

2）将 $z_i$ 的各分量载波到对应变量的取值区间.

3）计算粒子的适应值，并从 $N$ 个初始群体中选性能较好的 $M$ 个为初始解，随机产生 $M$ 个初始速度.

Step3：令粒子 $i$ 当前的最优位置为 $p_i=z_i$，对应的适应度为 $f_{\text{best}_i}=f_i$. 并从中找出全局最优粒子，令其位置为 $p_g$，对应的适应度为 $g_{\text{best}}$.

Step4：按式(9)和(10)修正 $\Delta_{id}^{1}$ 和 $\Delta_{id}^{2}$，并将其限制在 $[\Delta_{\min}^{d},\Delta_{\max}^{d}]$ 内.

Step5：按式(8)计算 $\Delta v_{id}^{1}$ 和 $\Delta v_{id}^{2}$.

Step6：对所有粒子执行如下操作：

1）按式(7)修正粒子飞行速度，将其限制在 $[-V_{d\max},V_{d\max}]$ 内；

2）按式(2)修正粒子位置，将其限制在 $[X_{\min},X_{\max}]$ 内，并计算适应度 $f_i$；

3）如果 $f_i<f_{\text{best}}$，则令 $p_i=z_i$，$f_{\text{best}}=f_i$；

4）如果 $f_i<g_{\text{best}}$，则令 $p_g=z_i$，$g_{\text{best}}=f_i$.

Step7：在最优位置 $P_g=(p_{g1},p_{g2},\cdots,p_{gD})$ 上进行混沌优化. 将 $p_{gi}(i=1,2,\cdots,D)$ 映射到 $[-1,1]$，然后用逻辑自映射方程迭代产生混沌序列 $z_i^{(m)}(m=1,2,\cdots)$，再将产生的混沌序列通过逆映射返回到原解空间可得

$$p_g^{(m)}=(p_{g1}^{(m)},p_{g2}^{(m)},\cdots,p_{gD}^{(m)}),\ m=1,2,\cdots.$$

计算其适应值可得到性能最好的可行解 $p^*$.

Step8：用 $p^*$ 取代当前群体中任一粒子的位置.

Step9：若满足停止条件，则搜索停止，并输出全局最优位置；否则，执行 Step10.

Step10：根据粒子适应值不同采取自适应策略，分别按式(6)调整 $w$，转 Step4.

## 4　仿真研究与分析

为了测试 CRPSO 算法的优化性能，本文利用 Rastrigrin 和 Rosenbrock 测试函数进行仿真研究，并将 CRPSO 算法与 CPSO 算法进行比较.

两个典型的测试函数为

$$f_1(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10],$$

$$f_2(x)=\sum_{i=1}^{9}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2].$$

其中：Rastrigrin 函数 $f_1$ 是很难最小化的病态二次函数，它具有一个全局极小点 $x^*=(0,0,\cdots,0)$，函数值为 $f(x^*)=0$；Rosenbrock 函数 $f_2$ 是具有大量局部最优点的多峰函数，它也有一个全局极小点 $x^*=(1,1,\cdots,1)$，函数值为 $f(x^*)=0$. 两个函数的变量范围都是 $[-5.12,5.12]$.

分别利用两种算法对上述函数进行仿真研究，种群大小均为 80，每种算法取 3 组不同的参数. 其中：pop = 80，$\Delta(0)\in[0.01\Delta u,0.1\Delta u]$，$\theta_+=1.1$，$\theta_-=0.5$，$c_1=c_2=2.0$，$\Delta_{\max}=0.5\Delta u$，$\Delta_{\min}=0.001\Delta u$，$w$ 的选取见上文，每组参数进行 20 次优化运算. 图 1 和图 2 分别是 CRPSO 算法与 CPSO 算法的优化性能比较.

由仿真结果可知，针对上面测试函数，CPSO 算法难以找出它们的全局最优解. 在算法前期，从总体

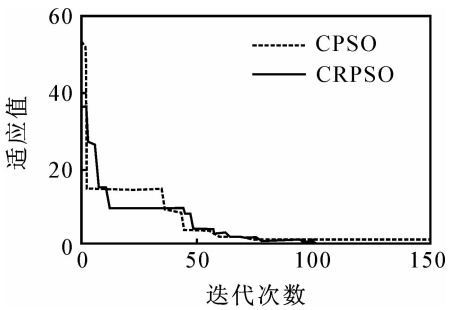


**图 1　$f_1$ 的性能指标变化曲线比较图**

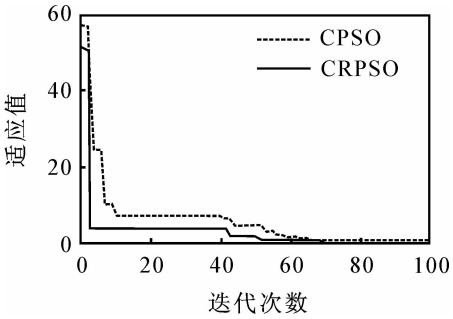


**图 2 $f_2$ 的性能指标变化曲线比较图**

上看目标函数值,CRPSO 算法较 CPSO 算法下降得快.这说明初始种群采用逻辑自映射函数来产生的混沌序列更具均匀性和更好的遍历性.在算法中后期,CRPSO 算法比 CPSO 算法的收敛速度更快,并且能收敛到目标函数的理论全局极值.由此验证了在算法后期,引入混沌序列的搜索算法可在迭代中产生局部最优解的许多邻域点,从而提高了种群的多样度,而且直接促成了 CRPSO 算法较 CPSO 算法具有更强跳出局部极小的能力,即能有效地避免算法陷入局部最优.对上述几个多极点函数的数值仿真,CRPSO 算法几乎次次能收敛到全局最优值.显然,CRPSO 算法比 CPSO 算法的优化效果更好.

## 5 结 论

本文分析了粒子群优化早熟收敛的原因,采用弹性改变粒子速度的策略,将混沌思想和弹性粒子群优化有机结合.同时,采用了基于早熟收敛评价的自适应改变惯性权重策略,从多个角度提高种群的多样度,有效避免了算法的早熟收敛,从而提高了算法的全局收敛性和收敛效率.成功地将改进算法用于多极点函数优化,但较多的参数给算法带来了不便,自适应修正参数将是下一步的研究内容.

## 参考文献(References)

[1] Kennedy J, Eberhartr C. Particle swarm optimization [C]. Proc of IEEE Int Conf on Neural Networks. Perth: IEEE Piscataway, 1995: 1942-1948.

[2] Kennedy J, Eberhart R. Swarm intelligence[M]. San Francisco: Morgan Kaufmann Publishers, 2001.

[3] 吴晓军, 薛会锋, 李憋, 等. GA-PSO 混合规划算法[J]. 西北大学学报, 2005, 35(1): 39-43.
(Wu X J, Xue H F, Li M, et al. A new programming of mixed genetic algorithm with particle swarm optimization[J]. J of Northwest University, 2005, 35(1): 39-43.)

[4] 支成秀, 梁正友. 融合粒子群优化算法与蚁群优化算法的随机搜索算法[J]. 广西科学院学报, 2006, 22(4): 231-233.
(Zhi C X, Liang Z Y. A stochastic searching algorithm in combination with particle swarm optimization algorithm and ant colony algorithm[J]. J of Guangxi Academy of Sciences, 2006, 22(4): 231-233.)

[5] 张劲松, 李歧强, 王朝霞. 基于混沌搜索的混合粒子群优化算法[J]. 山东大学学报, 2007, 37(1): 47-50.
(Zhang J S, Li Q Q, Wang Z X. Hybrid particle swarm optimization algorithm based on the chaos search[J]. J of Shandong University Engineering Science, 2007, 37(1): 47-50.)

[6] 李勇刚,桂卫华, 阳春华, 等. 一种弹性粒子群优化算法[J]. 控制与决策, 2008, 23(1): 95-98.
(Li Y G, Gui W H, Yang C H, et al . A resilient particle swarm optimization algorithm[J]. Control and Decision, 2008, 23(1): 95-98.)

(上接第 1544 页)

[9] Coughlan A T. Competition and cooperation in marketing channel choice: Theory and application[J]. Marketing Science, 1985, 4(2): 110-129.

[10] Moorthy K S. Decentralization in channels [J]. Marketing Science, 1988, 7(4): 335-355.

[11] Trivedi M. Distribution channels: An extension of exclusive retailership[J]. Management Science, 1998, 44(7): 231-246.

[12] Albert Y Ha, Shilu Tong. Contracting and information sharing under supply chain competition [J]. Management Science, 2008, 54(4): 701-715.

[13] 艾兴政, 唐小我. 基于讨价还价能力的竞争供应链纵向结构绩效研究[J]. 管理工程学报, 2007, 21(2): 123-125.
(Ai X Z, Tang X W. Performance of vertical structure under competitive supply chains based on bargaining power[J]. J of Management Engineering, 2007, 21(2): 123-125.)

[14] Wu Q, Chen H. Chain-chain competition under demand uncertainty[D]. Columbia: The University of British Columbia, 2003.

[15] 艾兴政, 唐小我,涂智寿. 不确定性环境下链与链竞争的纵向控制结构绩效[J]. 系统工程学报, 2008, 23(2): 188-193.
(Ai X Z, Tang X W, Tu Z S. Performance of vertical control structure of chain to chain competition under uncertainty[J]. J of System Engineering, 2008, 23(2): 188-193.)

[16] 艾兴政, 廖涛, 唐小我. 链与链竞争的充分退货政策[J]. 系统工程学报, 2008, 23(6): 727-734.
(Ai X Z, Liao T, Tang X W. Full return policies of chain to chain competition [J]. J of System Engineering, 2008, 23(6): 727-734.)

[17] Atkins D, Zhao X. Supply chain structure under price and service competition[D]. Columbia: The University of British Columbia, 2003.