doi:10.3969/j.issn.1000-436x.2014.04.015

自适应的分数阶达尔文粒子群优化算法

郭通，兰巨龙，李玉峰，陈世文
(国家数字交换系统工程技术研究中心，河南 郑州 450002)

摘 要：针对分数阶达尔文粒子群算法收敛性能依赖于分数阶次 α，易陷入局部最优的特点，提出了一种自适应的分数阶达尔文粒子群优化（AFO-DPSO）算法，利用粒子的位置和速度信息来动态调整分数阶次 α，并引入自适应的加速系数控制策略和变异处理机制，以获取更优的收敛性能。对几种典型函数的测试结果表明，相比于现有的粒子群算法，所提的 AFO-DPSO 算法的搜索精度、收敛速度和稳定性都有了显著提高，全局寻优能力得到了进一步提高。

关键词：分数阶达尔文粒子群优化；进化因子；分数阶次；加速系数；变异机制；自适应
中图分类号：TP301.6 **文献标识码**：A **文章编号**：1000-436X(2014)04-0130-11

Adaptive fractional-order Darwinian particle swarm optimization algorithm

GUO Tong, LAN Ju-long, LI Yu-feng, CHEN Shi-wen
(National Digital Switching System Engineering & Technological Research Center, Zhengzhou 450002, China)

Abstract: The convergence performance of the fractional-order Darwinian particle swarm optimization (FO-DPSO) algorithm depends on the fractional-order α, and it can easily get trapped in the local optima. To overcome such shortcoming, an adaptive fractional-order Darwinian particle swarm optimization (AFO-DPSO) algorithm was proposed. In AFO-DPSO, both particle's position and velocity information were utilized adequately, together an adaptive acceleration coefficient control strategy and mutation processing mechanism were introduced for better convergence performance. Testing results on several well-known functions demonstrate that AFO-DPSO substantially enhances the performance in terms of convergence speed, solution accuracy and algorithm stability. Compared with PSO, HPSO, DPSO, APSO, FO-PSO, FO-DPSO and NCPSO, the global optimality of AFO-DPSO are greatly improved.

Key words: fractional-order Darwinian particle swarm optimization; evolution factor; fractional-order; acceleration coefficients; mutation mechanism; adaptive

1 引言

粒子群优化（PSO, particle swarm optimization）算法是一种基于群体智能的随机全局优化计算方法[1]。该算法以其建模简单、收敛速度快且易于实现等优点，在组合优化[2]、多目标辨识[3]、任务分配[4]、聚类分析[5]、神经网络训练[6]等领域得到了广泛的应用。类似于其他全局优化算法，PSO 算法在实际应用中也表现出了一些有待改进的问题，即算法在搜索初期往往收敛较快，但在后期粒子群会趋向同一化(失去了多样性)，使得收敛速度明显变慢，搜索精度降低，算法容易陷入局部最优，因此很多学者致力于提高 PSO 算法的性能。

Naka 等[7]将遗传算法中的选择操作引入到 PSO 中，提出了混合粒子群优化(HPSO, hybrid particle swarm optimization)算法，对每次迭代产生的新的粒

收稿日期：2012-12-08；**修回日期**：2013-05-07
基金项目：国家重点基础研究发展计划（“973”计划）基金资助项目（2012CB315900）；国家高技术研究发展计划（“863”计划）基金资助项目（2011AA01A103）
Foundation Items: The National Basic Research Program of China (973 Program)(2012CB315900); The National High Technology Research and Development Program of China (863 Program)(2011AA01A103)

子群依照一定选择率复制较优个体，在提高收敛速度的同时保证了一定的全局搜索能力。Krohling[8]将高斯函数引入 PSO 算法中，用于引导粒子的运动，进而提出了高斯粒子群优化（GPSO, Gaussian particle swarm optimization）算法，GPSO 不再需要惯性权重，且加速系数由服从高斯分布的随机数产生，以克服传统 PSO 搜索能力和收敛性能严重依赖加速系数和惯性权重设置的不足。Jason 等[9]提出了一种利用自然选择进化思想的达尔文粒子群优化（DPSO, Darwinian particle swarm optimization）算法，动态地将种群分为若干个子群，每个子群相对独立地展开搜索，以提高粒子的多样性，增强算法的全局寻优能力。Xu 等[10]提出了一种新的混沌粒子群优化（NCPSO, new chaos-particle swarm optimization）算法，将混沌融入到粒子运动过程中，使粒子群在混沌与稳定之间交替向最优点靠近，能够跳出局部最优，提高了算法的收敛速度和精度。

作为应用自然科学中的一种有用的数学工具，分数阶微积分（FOC, fractional order calculus）理论在水文建模、统计力学、图像分割、模式识别、信号处理等实际工程领域的应用越来越广泛[11~14]。最近，Pires 等[15]将分数阶微积分引入到 PSO 中，提出了分数阶粒子群优化（FO-PSO）算法，通过对粒子群的原始速度进行重新排列来修正速度导数的阶数，以便于控制算法的收敛率。在此基础上，Micael 等[16]给出了一种分数阶达尔文粒子群优化（FO-DPSO）算法，用于控制 DPSO 算法的收敛速度，实验结果表明，FO-DPSO 算法在计算精度和收敛速度上均要优于传统的 PSO、DPSO 以及 FO-PSO 算法。但同 FO-PSO 算法一样，FO-DPSO 算法的收敛性能也直接依赖于分数阶次 α，当 α 值增加时，粒子群的收敛速度就变慢，而当 α 值减小时，种群陷入局部最优的概率就变高。

针对此不足，本文受文献[17]提出的自适应粒子群优化（APSO, adaptive particle swarm optimization）算法的启发，并在此基础上改进，结合 FO-DPSO 算法，提出了一种自适应的分数阶达尔文粒子群优化（AFO-DPSO）算法。对几种典型函数的测试结果表明，相比于现有的粒子群优化算法，本文的 AFO-DPSO 算法能够避免陷入局部最优，收敛速度、稳定性和精度都有了显著提高，进一步提高了全局寻优能力。

2　PSO 和 DPSO 算法概述

2.1　PSO 算法基本原理

设在一个 D 维的目标搜索空间中，有 N 个粒子组成一个群落，其中，第 i 个粒子的位置表示为向量 $X_i=(x_{i1},x_{i2},\cdots,x_{iD})$，速度为向量 $V_i=(v_{i1},v_{i2},\cdots,v_{iD})$，第 i 个粒子“飞行”历史中的过去最优位置为 $P_i=(p_{i1},p_{i2},\cdots,p_{iD})$，整个粒子群搜索到的最优位置 $P_g=(p_{g1},p_{g2},\cdots,p_{gD})$。将 X_i 代入目标函数计算其适应值，每个粒子的速度和位置更新策略分别为

$$v_{id}(t+1)=wv_{id}(t)+c_1r_{1d}\left(p_{id}(t)-x_{id}(t)\right)+c_2r_{2d}\left(p_{gd}(t)-x_{id}(t)\right) \tag{1}$$

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

其中，i=1,2,…,N；d=1, 2,…, D；在式(1)中，w 是惯性加权；c_1、c_2 是加速系数；r_{1d}、r_{2d} 为[0,1]之间的随机数；p_{id} 为粒子本身所找到的最优解，即个体极值；p_{gd} 是整个粒子群目前找到的最优解，称之为全局极值。应用式(1)和式(2)对种群中每个粒子群循环更新，可使整个种群逐步逼近全局最优解。

2.2　DPSO 算法基本思想

DPSO 是一种进化算法，其基本思想[9]如下所示。

1) 一个粒子群存在的时间越长，它产生后代（构造一个新的粒子群）的机会也就越大，当其产生后代时，后代还会继承一些适应度较好的粒子。

2) 一个粒子群可以通过找到更好的适应值来延长其寿命，同时还可以增加新的粒子来加快搜索速度。

3) 粒子群如果没能找到更好的适应值，它的寿命将会缩短，同时也可能删除群中那些性能较差的粒子，当粒子的数量减少到一定阈值时，该粒子群被销毁。

在 DPSO 算法中，为了追踪粒子群适应值没有改变的次数，设置一个搜索计数器为 SC，如果搜索计数器超过最大门限 $SC_c^{\max}$，将该粒子从群中删除，而创建一个粒子群时，SC 置为 0。在删除粒子后，SC 值并不置为 0，而是按照式(3)重置为一个接近 $SC_c^{\max}$ 的值。

$$SC_c(N_{\text{kill}})=SC_c^{\max}\left[1-\frac{1}{N_{\text{kill}}+1}\right] \tag{3}$$

其中，N_{kill} 为在适应值没有任何改善的一段时期内

从粒子群中所删除的粒子数目。在产生一个新的粒子群之前，粒子群中必须没有任何粒子被删除，并且粒子群数量不能超过其最大数目。尽管如此，创建新的粒子群概率也仅仅为

$$p = f / N_s \tag{4}$$

其中，f为[0,1]范围内的随机数，N_s为粒子群数目。

3 自适应分数阶达尔文粒子群优化算法

由文献[16]可知，在绝大多数情形下，当分数阶次 α 在[0.5,0.8]范围内时，算法能够获得更快的收敛速度，并给出了如下的表达式来调整 α。

$$\alpha(t) = 0.8 - 0.3 \times \frac{t}{200} \tag{5}$$

其中，t 为当前迭代次数，算法最大迭代次数为 200。

然而，式(5)中的 α 却会随着迭代次数的增加而线性减小，这无疑增大了种群陷入局部最优的概率。为了能有效避免算法的早熟收敛问题，同时能加快算法的收敛速度，本文提出一种根据粒子的状态信息来动态调整分数阶次 α 的自适应控制策略，并在算法中引入变异操作的处理方式，在此基础上给出了自适应的分数阶达尔文粒子群优化（AFO-DPSO）算法。

3.1 分数阶次 α 的自适应控制

对于每个粒子 i，它与其他粒子的平均距离与平均速率差异分别采用式(6)和式(7)来估计。

$$d_{ix} = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_{ik} - x_{jk})^2} \tag{6}$$

$$d_{iv} = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (v_{ik} - v_{jk})^2} \tag{7}$$

其中，N 和 D 分别为粒子群个数和空间维数，d_{ix} 代表平均距离，d_{iv} 表示平均速率差异。

众所周知，PSO 区别于其他进化算法的重要标志就是粒子的进化状态由位置和速度来共同描述，综合式(6)和式(7)，定义一种混合了粒子平均距离与平均速率差异信息的平均进化状态差异 d_{is} 如下所示。

$$d_{is} = d_{ix} + \rho_{X_i V_i} d_{iv} \tag{8}$$

$$\rho_{X_i V_i} = \frac{\mathrm{E}(X_i V_i) - \mathrm{E}(X_i)\mathrm{E}(V_i)}{\sqrt{\mathrm{E}(X_i^2) - \mathrm{E}^2(X_i)}\sqrt{\mathrm{E}(V_i^2) - \mathrm{E}^2(V_i)}}$$

$$= \frac{\sum_{j=1}^{D} x_{ij} v_{ij} - \frac{1}{D}\sum_{j=1}^{D} x_{ij} \sum_{j=1}^{D} v_{ij}}{\sqrt{\sum_{j=1}^{D} x_{ij}^2 - \frac{1}{D}\left(\sum_{j=1}^{D} x_{ij}\right)^2} \cdot \sqrt{\sum_{j=1}^{D} v_{ij}^2 - \frac{1}{D}\left(\sum_{j=1}^{D} v_{ij}\right)^2}} \tag{9}$$

其中，$\rho_{X_i V_i}$ 为位置 X_i 和速度 V_i 的皮尔逊相关系数（PCC, Pearson correlation coefficient）。

比较所有的 d_{is}，确定粒子进化状态的最大差异值 $d_{s\max}$ 和最小差异值 $d_{s\min}$。设 d_{sg} 为全局最优粒子与其他粒子的进化状态差异的平均值，定义进化因子 f_s 为

$$f_s = \frac{d_{sg} - d_{s\min}}{d_{s\max} - d_{s\min}} \in [0,1] \tag{10}$$

将 f_s 值通过模糊集映射函数将粒子进化状态分类为探测、开拓、收敛和跃出 4 种[17]。

基于进化因子 f_s，给出分数阶次 α 的调整等式为

$$\alpha(f_s) = \frac{1}{1 + \mathrm{e}^{-1.386\,3 f_s}} \in [0.5, 0.8], \forall f_s \in [0,1] \tag{11}$$

α 的变化不再与迭代次数相关，而是根据粒子进化状态信息的改变而进行动态调整。

3.2 加速系数自适应控制与粒子状态更新

将式(1)中惯性加权 w 设置为 1，依据分数阶微分的 Grünwald-Letnikov 定义，粒子速度按式(12)进行更新。

$$v_{t+1} = \alpha(f_s) v_t + \frac{1}{2}\alpha(f_s) v_{t-1} + \frac{1}{6}\alpha(f_s) v_{t-2} + \frac{1}{24}\alpha(f_s)(1-\alpha(f_s))(2-\alpha(f_s)) v_{t-3} + c_1 r_{1d}\left(p_{id}(t) - x_{id}(t)\right) + c_2 r_{2d}\left(p_{gd}(t) - x_{id}(t)\right) \tag{12}$$

其中，$\alpha(f_s)$即分数阶次。粒子位置按照式(2)进行更新，并根据表 1 中的规则来调整不同状态时的加速系数，且 c_1 和 c_2 满足

$$3 \leqslant c_1 + c_2 \leqslant 4 \tag{13}$$

表 1 中的“微”调采用式(14)来定量定义。

$$c_i(g+1) = c_i(g) \pm \delta, i = 1,2 \tag{14}$$

其中，δ 为[0.05, 0.1]范围内均匀产生的随机数。

表 1　速系数 c_1 和 c_2 的调整策略

状态	c_1	c_2
探测	增加	减少
开拓	微增	微减
收敛	微增	微增
跃出	减少	增加

3.3　基于进化状态的自适应变异处理机制

当粒子群陷入局部最优时，算法将出现早熟收敛。为了克服早熟收敛问题，ALFI[18]在对动态系统的参数进行估计时提出了一种采用自适应变异机制的 PSO 算法；Chen 等[19]在 PID 控制参数的优化设计中给出了一种变异概率随算法迭代次数逐步减少的新 PSO 算法。当算法发生早熟收敛时，变异操作可以改变粒子的前进方向，从而让粒子进入其他区域进行搜索，直到最后找到全局最优解，使得算法能够跳出局部最优。

借鉴文献[18]和文献[19]的思想，本文设计了一种随粒子进化状态动态变化的变异算子，根据变异概率 p_m 对满足变异条件的全局最优位置 P_g 实施变异。

定义 1　p_m 由进化因子 f_s 和迭代次数 t 共同表示，其选取遵照以下基本原理：

1) 进化状态为探测与开拓阶段时，t 占主导地位；

2) 进化状态为收敛和跃出阶段时，f_s 占主导地位；

3) $p_m(t, f_s)$为一致凸函数。

为给出直观的解释，按照定义 1 选择了一个特殊的 $p_m(t, f_s)$函数，其计算公式为

$$p_m\left(t,f_s\right)=1-\frac{t}{\left(1+t\right)^{f_s+1}-\left(1+t\right)^{f_s}} \tag{15}$$

对于全局最优位置 P_g 的变异操作，本文算法将采用增加高斯随机扰动的方法，设 p_{gk} 为 P_g 的第 k 维取值，η 是服从 Gauss(0,1)分布的随机变量，则

$$p_{gk}=p_{gk}\left(1+\eta\tanh(f_s)\right) \tag{16}$$

由于 $0\leqslant f_x\leqslant 1$，$0\leqslant\tanh(\cdot)\leqslant 1$，由此可以推断出 $1\leqslant 1+\eta\tanh(f_x)\leqslant 2$，$P_g$ 变异后的第 k 维取值随进化因子在$\left[p_{gk},2p_{gk}\right]$范围内变化。

3.4　算法的收敛性分析

令 $\phi_1=c_1r_{1d},\phi_2=c_2r_{2d},y=p_{id}(t),\hat{y}=p_{gd}(t)$，综合式(12)和式(2)，可得

$$\begin{aligned}x(t+1)=&x(t)+\alpha(f_s)v(t)+\frac{1}{2}\alpha(f_s)v(t-1)+\frac{1}{6}\alpha(f_s)v(t-2)+\\&\frac{1}{24}\alpha(f_s)\left(1-\alpha(f_s)\right)\left(2-\alpha(f_s)\right)v(t-3)+\\&\phi_1\left(y-x(t)\right)+\phi_2\left(\hat{y}-x(t)\right)\end{aligned} \tag{17}$$

由式(2)可知

$$x(t)=x(t-1)+v(t) \tag{18}$$

因此，

$$v(t)=x(t-1)-x(t) \tag{19}$$

将式(19)代入式(17)，可得

$$\begin{aligned}x(t+1)=&\left(1+\alpha(f_s)-\phi_1-\phi_2\right)x(t)+\phi_1y+\\&\phi_2\hat{y}-\frac{1}{2}\alpha(f_s)x(t-1)-\frac{1}{3}\alpha(f_s)x(t-2)-\\&\left(\frac{1}{6}\alpha(f_s)-\frac{\alpha(f_s)}{24}\left(1-\alpha(f_s)\right)\left(2-\alpha(f_s)\right)\right)\cdot\\&x(t-3)-\frac{\alpha(f_s)}{24}\left(1-\alpha(f_s)\right)\left(2-\alpha(f_s)\right)x(t-4)\end{aligned} \tag{20}$$

式(20)可以写成如下矩阵向量乘积形式。

$$\begin{bmatrix}x(t+1)\\x(t)\\x(t-1)\\x(t-2)\\x(t-3)\\1\end{bmatrix}=\begin{bmatrix}1+\alpha-\phi_1-\phi_2 & -\frac{1}{2}\alpha & -\frac{1}{3}\alpha & -\left(\frac{1}{6}\alpha-\frac{\alpha}{24}(1-\alpha)(2-\alpha)\right) & -\frac{\alpha}{24}(1-\alpha)(2-\alpha) & \phi_1y+\phi_2\hat{y}\\1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}x(t)\\x(t-1)\\x(t-2)\\x(t-3)\\x(t-4)\\1\end{bmatrix}$$

矩阵的特征多项式为

$$\begin{aligned}&\left[\lambda^4\left(1+\alpha-\phi_1-\phi_2\right)-\frac{1}{2}\alpha\lambda\left(\lambda^2+1\right)+\right.\\&\left.\frac{\alpha}{24}(1-\alpha)(2-\alpha)(\lambda+1)\right](1-\lambda)\end{aligned} \tag{21}$$

该特征多项式的一个解为 λ=1.0，令其他 4 个解分别为 β、γ、δ 和 ε，则给出式(17)的当前关系式为

$$x(t)=k_1+k_2\beta^t+k_3\gamma^t+k_4\delta^t+k_5\varepsilon^t \tag{22}$$

其中，k_1、k_2、k_3、k_4 和 k_5 均为常数。

考虑 $x(t)$的收敛性，即

$$\lim_{t\to\infty} x(t)=\lim_{t\to\infty}(k_1+k_2\beta^t+k_3\gamma^t+k_4\delta^t+k_5\varepsilon^t) \quad (23)$$

明显地，只要 β、γ、δ 和 ε 满足 max(||β||, ||γ||, ||δ||, ||ε||)<1，则

$$\lim_{t\to\infty} x(t)=\lim_{t\to\infty}(k_1+k_2\beta^t+k_3\gamma^t+k_4\delta^t+k_5\varepsilon^t)=k_1 \quad (24)$$

只需证明在任意 max(||β||, ||γ||, ||δ||, ||ε||)>1 条件下，特征多项式(21)均不成立即可。假设||β||>1，代入式(21)可得

$$\left[\|\beta\|^4(1+\alpha-\phi_1-\phi_2)-\frac{1}{2}\alpha\beta\left(\|\beta\|^2+1\right)+\frac{\alpha}{24}(1-\alpha)(2-\alpha)(\beta+1)\right](1-\beta) \quad (25)$$

由于 r_{1d}, r_{2d}在[0,1]范围内，$3\leqslant c_1+c_2\leqslant 4$，$\alpha(f_s)$在[0.5,0.8]区间内，$\frac{\alpha}{24}(1-\alpha)(2-\alpha)$ 为单调递减函数，故其最大值为 $\frac{1}{64}$，$-2.5\leqslant 1+\alpha-\phi_1-\phi_2\leqslant -1.2$，令

$$z=\left[\|\beta\|^4(1+\alpha-\phi_1-\phi_2)-\frac{1}{2}\alpha\beta\left(\|\beta\|^2+1\right)+\frac{\alpha}{24}(1-\alpha)(2-\alpha)(\beta+1)\right](1-\beta)$$

则

$$0<-0.4+0.4\|\beta\|\leqslant z\leqslant -2+2\|\beta\| \quad (26)$$

这就与||β||>1 为 z=0 的解相矛盾，故假设不成立，β、γ、δ 和 ε 始终满足 max(||β||, ||γ||, ||δ||, ||ε||)<1。

因此，本文提出的 AFO-DPSO 算法在给定的参数范围内是全局收敛的。

3.5 AFO-DPSO 算法描述

AFO-DPSO 算法的实现步骤可概括如下。

step1 初始化粒子群，确定粒子数 N 及空间维数 D，设定最大迭代代数 $t_{\max}$、加速系数 c_1 和 c_2，指定位置和速度的上限及下限，随机生成粒子速度 V_i 和位置 X_i。

step2 对每个粒子进行适应度评估，将粒子的个体极值 p_{id} 设置为当前位置，全局极值 p_{gd} 设置为初始群体中最佳粒子的位置。

step3 判断算法收敛准则是否满足，如果满足，转向 step8；否则，执行 step4。

step4 开始迭代，进化代数增加 1。

step5 对粒子群中所有粒子执行如下操作：

① 根据式(6)~式(9)得到粒子的平均进化状态差异，根据式(10)获得进化因子，在此基础上，应用式(11)更新分数阶次；

② 根据表 1 中规则和式(13)、式(14)调整加速系数，在上述参数更新的基础上，应用式(12)与式(2)实现粒子位置与速度的更新；

③ 对更新状态后粒子的适应度值进行评估，如果粒子适应度优于当前的个体极值，则将 p_{id} 设置为该粒子的位置，且更新个体极值；如果所有粒子的个体极值中最好的优于当前全局极值，则将 p_{gd} 设置为该粒子的位置，且更新全局极值。

step6 对粒子群的适应度值进行追踪，判断是否需要删除粒子或创建新的粒子群，若删除粒子，则根据式(3)重置搜索计数器值，若构造新的粒子群，则根据式(4)得到产生粒子群的概率；否则，转向 step7。

step7 根据式(15)计算变异概率 p_m。每个粒子设定一个(0,1)间随机数 rnd_i，若 $rnd_i<p_m$，按式(16)执行变异操作；否则，转向 step8。

step8 判断是否满足算法的收敛条件或达到代数最大限制，如果满足，执行 step9；否则，转向 step4。

step9 输出最终的全局极值 p_{gd} 与最佳适应度函数值，算法运行结束。

上述算法既继承了 FO-DPSO 算法的优点，同时又能根据粒子的状态信息来自适应地调整分数阶次 α，从而加快了算法的收敛速度，并降低了种群陷入局部最优的概率。而自适应的变异处理方式在提高粒子多样性的同时也增强了算法的全局寻优能力。

4 实验结果与分析

4.1 测试函数与算法配置

为了考察 AFO-DPSO 算法的性能，选取 6 个典型函数进行测试，它们是在群智能优化算法中广泛采用的测试函数(求最小值)，即 Sphere、Rosenbrock、DeJong F4、Rastrigin、Griewank 和 Ackley 函数，它们的表达式如式(27)~式(32)所示[20]。

Sphere 函数

$$f_1(x)=\sum_{i=1}^{D}x_i^2 \quad (27)$$

其中，$x_i\in[-50,50]$，$i=\{1,2,\cdots,D\}$且 $f^*(x)=0.0$。

Rosenbrock 函数

$$f_2(x)=\sum_{i=1}^{D}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right) \tag{28}$$

其中，$x_i\in[-100,100]$，$i=\{1,2,\cdots,D\}$且$f^*(x)=0.0$。

DeJong F4 函数

$$f_3(x)=\sum_{i=1}^{D}x_i^4 \tag{29}$$

其中，$x_i\in[-20,20]$，$i=\{1,2,\cdots,D\}$且$f^*(x)=0.0$。

Rastrigin 函数

$$f_4(x)=\sum_{i=1}^{D}\left(x_i^2-10\cos\left(2\pi x_i\right)+10\right) \tag{30}$$

其中，$x_i\in[-5.12,5.12]$，$i=\{1,2,\cdots,D\}$且$f^*(x)=0.0$。

Griewank 函数

$$f_5(x)=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \tag{31}$$

其中，$x_i\in[-600,600]$，$i=\{1,2,\cdots,D\}$且$f^*(x)=0.0$。

Ackley 函数

$$f_6(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right)+20+e \tag{32}$$

其中，$x_i\in[-32,32]$，$i=\{1,2,\cdots,D\}$且$f^*(x)=0.0$。

这些函数有 D=30 个参数，且它们的全局极值为 f^*。在这些函数中，f_1，f_2，f_3 为单峰函数，而 f_4，f_5，f_6 为多峰函数，算法采用实数编码方案。

本文中所用的仿真软件为 MATLABR2008b，仿真环境：CPU 为 Intel Pentium 4 3.20 GHz，内存为 2 GB，操作系统为 Microsoft Windows XP Professional SP2。选取 PSO、HPSO、DPSO、APSO、FO-PSO、FO-DPSO、NCPSO 算法与本文提出的 AFO-DPSO 算法进行比较，并将式(27)~式(32)作为适应度函数。

各 PSO 算法的初始参数设置如表 2 所示。为了确保测试的公平性，所有 PSO 均采用：粒子数 N=30，空间维数 D 为 30，最大迭代次数 $t_{\max}$=1 000。

4.2　搜索精度比较

为了减少统计误差，采用各 PSO 算法对每个函数进行 30 次测试，取其平均值的结果如表 3 所示，其中，APSO 算法结果来自文献[17]，而 NCPSO 算法结果来自文献[10]。

对比表 3 中的结果可以看出，无论是在单峰函数 f_1、f_2、f_3 上，还是在多峰函数 f_4、f_5、f_6 上，本文提出的自适应分数阶达尔文粒子群优化算法的测试结果均要明显好于现有的 PSO、HPSO、DPSO、APSO、FO-PSO、FO-DPSO 与 NCPSO 算法的测试结果，且本文算法在 f_5 上获得了全局最优值。

4.3　算法稳定性能比较

进一步地，本文引入文献[21]中定义的最优解方差函数来评价算法的稳定性能。设 f_i 为算法第 i 次运行所得的最优解适应度，f_{avg} 为算法运行 30 次的最优解的平均值，$f_{\max}$ 为归一化因子，算法最优解方差 δ 定义为

$$\delta=\sum_{i=1}^{30}\left|\frac{f_i-f_{\text{avg}}}{f_{\max}}\right|^2 \tag{33}$$

当 $|f_i-f_{\text{avg}}|>1$ 时，$f_{\max}$ 取值为 $\max(|f_i-f_{\text{avg}}|)$，否则取 1。表 4 列出了各 PSO 算法对 6 个测试函数

表 2　　**各 PSO 算法的参数设置**

算法	参数	参考文献
PSO	w: 0.9~0.4, c_1=c_2=2	[1]
HPSO	w: 0.8~0.2, c_1=c_2=2.5, $V_{\max d}$=0.5×*Range*	[7]
DPSO	$sc_{\max}$ = 15, sc = 0, w=0.9, N_{kill} = 0	[9]
APSO	自动选择	[17]
FO-PSO	w=0.9, α = 0.632, c_1=c_2=1.5, r_{1d}: 0~1, r_{2d}: 0~1	[15]
FO-DPSO	w=0.9, α = 0.632, c_1=c_2=1.5, $sc_{\max}$ = 15, sc = 0, min_*swarms*=4, max_*swarms*=8	[16]
NCPSO	w=0.729 8, c_1=c_2=1.496 2, r_{1d}: 0~1, r_{2d}: 0~1, $V_{\max d}$=*Range*	[10]
AFO-DPSO	w=1, c_1: 1.5~2.5, c_2: 1.5~2.5, δ: 0.05~0.1, r_{1d}: 0~1, r_{2d}: 0~1, $V_{\max d}$=*Range*	本文

表 3 不同粒子群优化算法对 6 个测试函数的搜索结果比较

优化算法	测试函数					
	Sphere	Rosenbrock	DeJong F4	Rastrigin	Griewank	Ackley
PSO	$3.700\,4\times10^{2}$	13.865	$4.346\,7\times10^{3}$	1.0655×10^{2}	2.61×10^{7}	11.495 3
HPSO	3.876×10^{-1}	3.236 3	6.35×10^{-2}	4.8642	2.37×10^{-2}	5.697 2
DPSO	3.28×10^{-2}	2.343×10^{-1}	$1.375\,2\times10^{-5}$	1.9899	7.4×10^{-3}	2.408 3
APSO	1.45×10^{-10}	2.84	2.13×10^{-10}	1.01	1.67×10^{-2}	3.55×10^{-1}
FO-PSO	$1.534\,0\times10^{-5}$	6.1×10^{-3}	$9.252\,1\times10^{-12}$	3.5×10^{-3}	$1.418\,5\times10^{-7}$	1.4×10^{-3}
FO-DPSO	$3.472\,8\times10^{-7}$	1.1×10^{-3}	$8.809\,8\times10^{-16}$	$4.230\,5\times10^{-5}$	$8.137\,7\times10^{-9}$	$1.377\,4\times10^{-6}$
NCPSO	$2.027\,9\times10^{-9}$	$2.906\,8\times10^{-4}$	$2.080\,9\times10^{-17}$	$4.374\,1\times10^{-4}$	9.905×10^{-11}	$9.086\,9\times10^{-7}$
AFO-DPSO	$2.342\,0\times10^{-14}$	$2.206\,9\times10^{-10}$	$6.336\,4\times10^{-23}$	$1.895\,6\times10^{-10}$	0	$3.661\,0\times10^{-11}$

表 4 测试函数的最优值方差比较

优化算法	测试函数					
	Sphere	Rosenbrock	DeJong F4	Rastrigin	Griewank	Ackley
PSO	2.097 8	2.454 8	1.796 0	0.348 8	1.640 8	0.907 4
HPSO	1.602 2	1.629 5	1.013 0	0.216 2	1.365 8	0.020 0
DPSO	1.006 8	1.071 8	0.961 1	0.077 4	0.937 1	0.016 2
APSO	0.512 6	0.625 5	0.838 1	0.017 3	0.681 9	0.001 1
FO-PSO	0.750 5	0.834 1	0.883 6	0.023 2	0.815 1	0.002 5
FO-DPSO	0.409 1	0.334 2	0.734 4	0.013 7	0.657 4	$5.905\,8\times10^{-4}$
NCPSO	0.059 7	0.083 7	0.276 5	0.004 3	0.191 2	$3.541\,6\times10^{-4}$
AFO-DPSO	0.003 1	0.008 4	0.020 1	0.001 7	0.011 6	$1.463\,7\times10^{-5}$

进行优化所得到的最优值方差结果。从中可以看出，本文提出的 AFO-DPSO 算法的最优值方差比其他 7 种方法均要小。这说明 AFO-DPSO 算法搜索到的最优解最稳定。

4.4 收敛速度比较

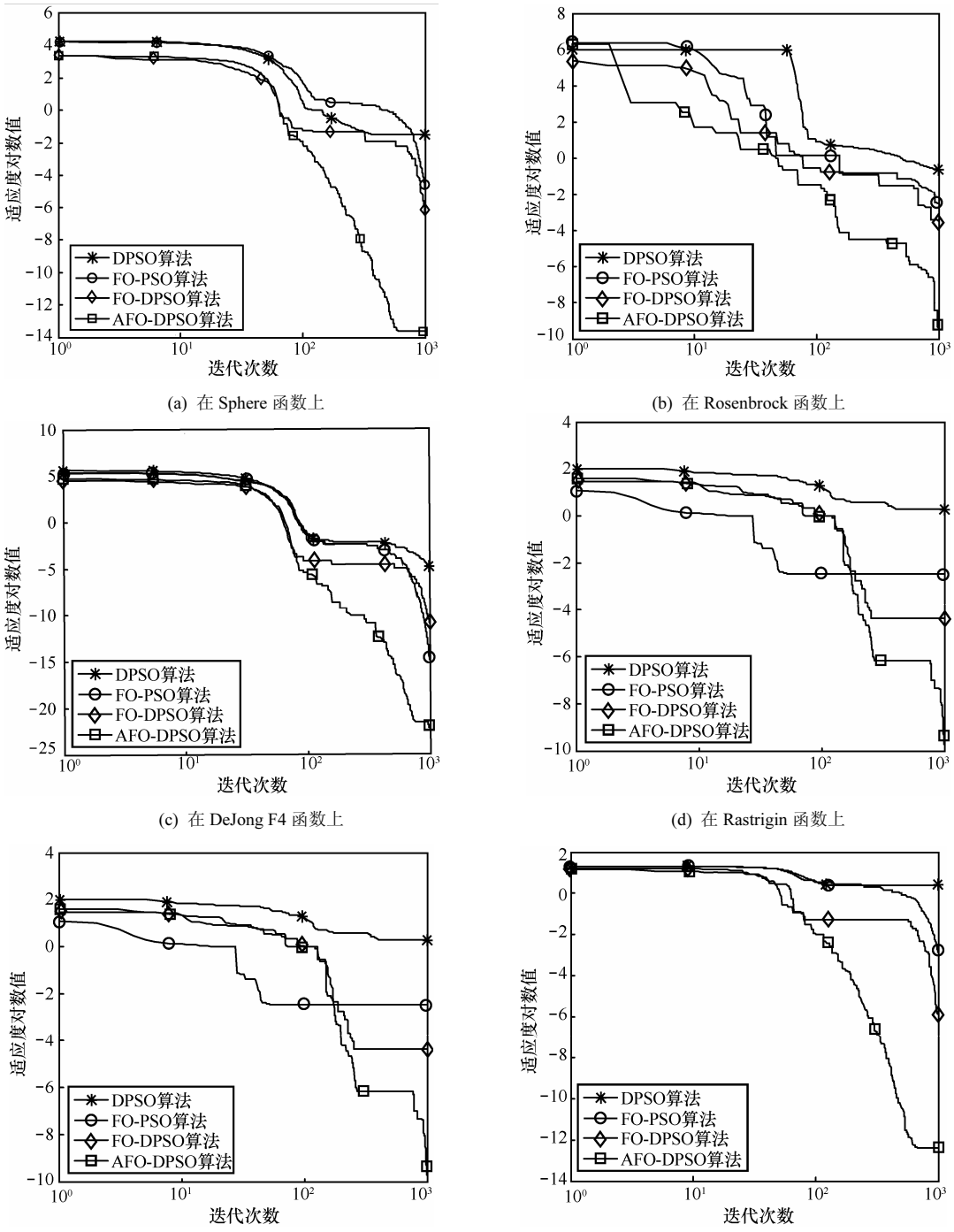
图 1 为现有 DPSO、FO-PSO、FO-DPSO 算法与本文提出的改进的分数阶粒子群优化算法对 6 个典型测试函数进行优化的过程中种群收敛性能的对比。为了便于比较，图 1 中的纵坐标都采用适应度的对数值表示。

由图 1 中 4 种不同粒子群算法在 6 个测试函数上的收敛性能比较可以看出，相比于 DPSO、FO-PSO 和 FO-DPSO 算法，本文提出的 AFO-DPSO 算法具有更快的全局收敛速度，粒子群在经过若干次迭代运算后仍具备从局部最优中跳离出来的能力，即使是在多模空间，由于采用了自适应参数调整策略，也能够快速地搜寻到潜在的最优极值，从而极大地提高了对最优值的全局搜索能力，并有效避免了现有粒子群优化算法中存在的早熟收敛问题。

综上所述，与现有的 7 种 PSO 算法相比，AFO-DPSO 算法展示出更令人满意的整体优化性能，其收敛速度、稳定性和搜索精度都得到了显著提高。

4.5 算法的计算复杂度分析

按照“天下没有免费的午餐（NFL, no free lunch）”理论[22]，一种算法不可能在每一个方面或在每一个问题上都能够提供比其他所有算法更好的性能。在本文的实验结果中也观察到了这点。考察 8 种算法在 1 000 次的迭代次数内对上文 6 个函数的平均计算复杂度，对各函数分别进行 30 次实



(a) 在 Sphere 函数上　　(b) 在 Rosenbrock 函数上

(c) 在 DeJong F4 函数上　　(d) 在 Rastrigin 函数上

(e) 在 Griewank 函数上　　(f) 在 Ackley 函数上

图 1　4 种粒子群算法在不同测试函数上的收敛性能比较

验，每种算法所需的平均计算时间如图 2 所示。

从图 2 中可以看出，AFO-DPSO 算法的算法复杂度要明显高于其他算法，FO-DPSO 算法与 DPSO 算法次之，其主要原因为：DPSO 算法需要根据粒子群的适应度值删除粒子或创建新的粒子群，导致算法运算时间过长，FO-DPSO 算法对粒子速度的分数阶运算进一步增加了时间复杂度，本文的 AFO-DPSO 算法由于利用了粒子的位置和速度信息来动态调整分数阶次，使得粒子每次更新都要增加 $O(2(N-1)D+N)$的计算复杂度。这表明，AFO-DPSO 算法具有的较高搜索精度、较快收敛速度和较好稳定性是以牺牲一定计算复杂度为代价获得的，并进一步验证了文献[22]中 NFL 理论的正确性。

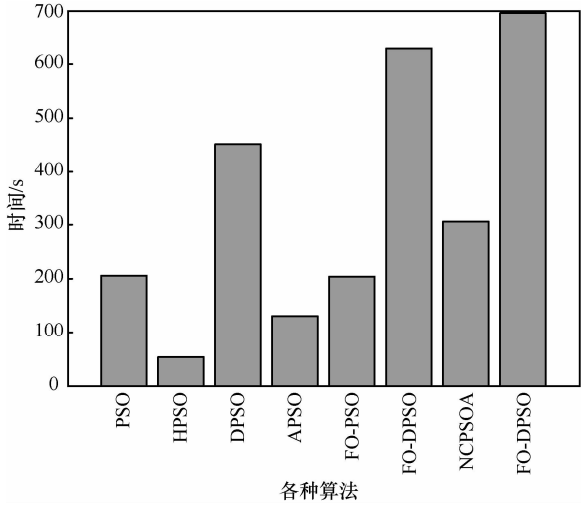


图 2 8 种不同算法的计算复杂度比较

4.6 算法参数的自适应性分析

为了观察进化因子 f_s 与分数阶次 α 的变化情况，在单峰函数 f_1 上运行 AFO-DPSO 算法，并绘制出随迭代次数变化的 f_s 值和 α 值，分别如图 3 与图 4 所示。从图 3 中可以看出，在 1 000 次的迭代次数中，进化因子 f_s 值在早期阶段（大约 10 代内）很大，然后迅速减少（直至 100 代），紧接着可以发现 AFO-DPSO 处于跃出状态，并导致 f_s 突变至一个较大值，在此之后，f_s 值迅速下降直至一个收敛状态（在 135 代时）；大约经过 150 代后，粒子群从该局部最优中跃离出来，期间（在 300~700 代内）经历多个“搜寻－收敛－跳离”阶段后，达到全局（1 000 代内）最优收敛状态。在整个进化过程中，f_s 值始终在[0,1]范围内变化，且较为明晰地展示出了粒子群的进化状态信息。

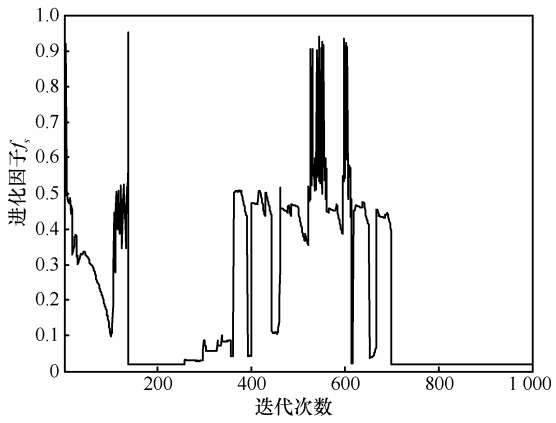


图 3 测试函数 f_1 上进化因子 f_s 所显示的进化状态信息

由图 4 中可以看到，分数阶次 α 值随着进化因子 f_s 的改变而在[0.5,0.8]区间内动态变化，这表明分数阶次 α 能够依据粒子群的进化状态进行自适应调整。为了验证算法的通用性，在函数 f_2~f_6 上重复这一实验，进化因子与分数阶次变化情况展现出与在函数 f_1 上类似的模式，在此不再赘述。

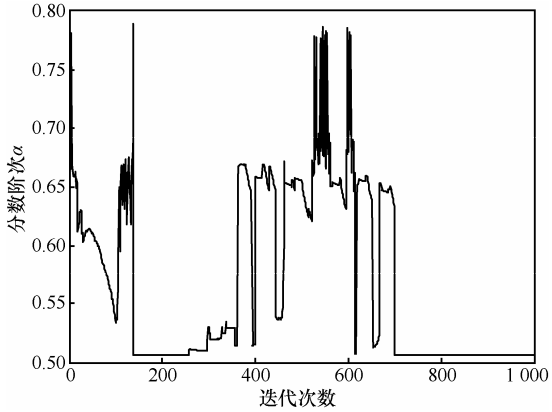


图 4 测试函数 f_1 上的时变分数阶次 α

为了追踪变异概率 p_m 的变化，类似地，图 5 给出了对 f_1 函数进行测试时，在 1 000 次的迭代次数内 p_m 的变化曲线。可以看出，在起始阶段由于粒子群在探测最优值，混合进化因子 f 值还较大，在一定代数内，AFO-DPSO 算法会维持一个大的变异概率 p_m；随着 f 的急剧减小，p_m 值降至接近于 0；当 f 发生跃变时，p_m 也随之产生一定的波动；至粒子群搜索达到收敛状态后，p_m 基本保持在 0 值附近。

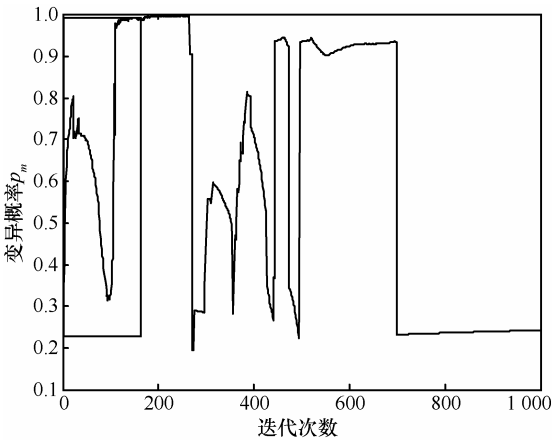


图 5 f_1 上变异概率 p_m 的变化曲线

为了追踪加速系数的变化，图 6 给出了在 f_1 上 1 000 次的迭代次数内 c_1 和 c_2 的变化曲线。

笔者发现：在起始阶段，由于粒子群在探测最优值，在一定代数内，c_1 在增加的同时 c_2 在减少；然后在开拓收敛状态时，c_1 和 c_2 倒转了它们的变化方向；跃出状态也能够被检测出来，其中，c_2 的值

在增加而 c_1 的值却在减少。

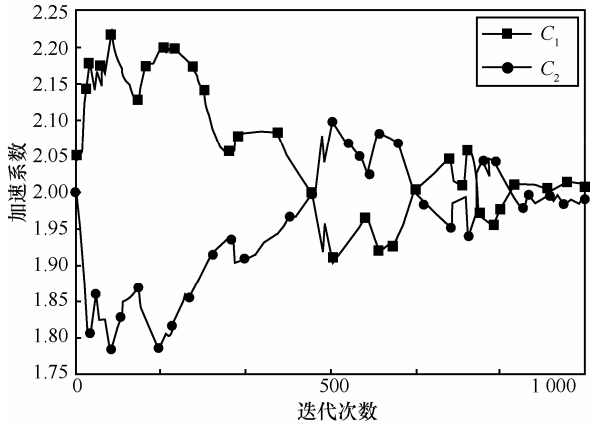


图 6　测试函数 f_1 上加速系数的自适应变化曲线

综上，进化因子、分数阶次、加速系数和变异概率的搜索行为表明经过改进后的 FO-DPSO 算法能够识别出粒子的进化状态，并能根据粒子的状态信息来自适应地调整控制参数。同时，这也说明本文提出的 AFO-DPSO 算法是有效且可行的。

5　结束语

为了能有效解决分数阶达尔文粒子群优化算法的收敛性能严重依赖于分数阶次 α 取值这一问题，同时进一步加快算法的收敛速度，本文提出一种根据粒子的状态信息来动态调整分数阶次 α 的自适应控制策略，并引入自适应的加速系数控制准则和变异处理机制，在此基础上给出了 AFO-DPSO 算法。对几种典型函数的测试结果表明，相比于现有的 PSO、HPSO、DPSO、APSO、FO-PSO、FO-DPSO 与 NCPSO 算法，本文的 AFO-DPSO 算法的搜索精度、收敛速度和稳定性都有了显著提高，全局寻优能力得到了极大提升。此外，对进化因子、分数阶次、加速系数和变异概率的搜索行为分析进一步验证了本文提出算法的有效性和可行性。下一步研究方向是如何确保收敛性能不变的同时降低 AFO-DPSO 算法的计算复杂度。

参考文献：

[1] KENNEDY J, EBERHART R. Particle swarm optimization[A]. Proc IEEE International Conf on Neural Networks[C]. 1995.1942-1948.

[2] CHEN, W N, ZHANG. A novel set-based particle swarm optimization method for discrete optimization problem[J]. IEEE Transactions on Evolutionary Computation, 2010, 14 (2): 278-300.

[3] ZHANG T, HU T S, ZHENG Y, GUO X N. An improved particle swarm optimization for solving bi-level mulitiobjective programming problem[J]. Journal of Applied Mathematics, 2012, 2(4): 1-13.

[4] HO S Y, LIN H S, LIAUH W H, *et al*. OPSO: Orthogonal particle swarm optimization and its application to task assignment problems[J]. IEEE Transactions on Systems, Man, Cybernetics A: Systems, Humans, 2008, 38(2):288-298.

[5] NIE F, TU T, PAN M, *et al*. K-Harmonic Means Data clustering with PSO Algorithm[M]. Springer Berlin Heidelberg, 2012.67-73.

[6] VARSHNEY S, SRIVASTAVA L, PANDIT M. Parameter tuning of statcom using particle swarm optimization based neural network[J]. Intelligent and Soft Computing, 2012, 130(3): 813-824.

[7] NAKA S, GENJI T, YURA T, *et al*. A hybrid particle swarm optimization for distribution state estimation[J]. IEEE Trans on Power Syst, 2003,18(1):60-68.

[8] KROHLING R. Gaussian particle swarm with jumps[A]. Proc IEEE Congr Evol Comput[C]. 2005.1226-1231.

[9] TILLETT J, RAO T, SAHIN F, *et al*. Darwinian particle swarm optimization[A]. Indian International Conference on Artificial Intelligence[C]. 2005.1474-1487.

[10] 胥小波, 郑康锋, 李丹等. 新的混沌粒子群优化算法[J]. 通信学报, 2012, 33(1): 24-31.
XU X B, ZHENG K F, LI D, *et al*. New chaos-particle swarm optimization algorithm[J]. Journal on Communications, 2012, 33(1): 24-31.

[11] GUTIÉRREZ R E, ROSÁRIO J M, MACHADO J T. Fractional order calculus: basic concepts and engineering applications[J]. Mathematical Problems in Engineering, 2010, 2010:1-19.

[12] MACHADO J A T, JESUS I S, BARBOSA R, *et al*. Application of fractional calculus in engineering[J]. Dynamics, Games and Science I, Springer Proceedings in Mathematics, 2011, 1: 619-629.

[13] BENSON D A, MEERSCHAERT M M, REVIELLE J. Fractional calculus in hydrologic modeling: a numerical perspective[J]. Water Resources, 2013,51:479-497.

[14] GHAMISI P, COUCEIRO M S, BENEDIKTSSON J A, *et al*. An efficient method for segmentation of images based on fractional calculus and natural selection[J]. Expert Systems with Applications, 2012, 39(16): 12407-12417.

[15] PIRES J A, MOURA P B, OLIVEIR A M, *et al*. Particle swarm optimization with fractional-order velocity[J]. Nonlinear Dynamics, 2010, 61(1-2): 295-301.

[16] COUCEIRO M S, ROCHA R P, FONSECA FERREIRA N M, *et al*. Introducing the fractional-order Darwinian PSO[J]. Signal, Image and Video Processing, 2012,6(3):343-350.

[17] ZHAN Z, ZHANG J, LI Y, *et al*. Adaptive particle swarm optimization[J]. IEEE Transactions on Systems, Man, Cybernetics B: Cybernetics, 2009,39(6):1362-1381.

[18] ALIREZA A. PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems[J]. Acta Automatic Sinica, 2011, 37(5):541-549.

[19] CHEN X, ZHANG Y. Optimum design of PID controller parameters by improved particle swarm optimization algorithm[J]. Computer Science and Information Engineering, Lecture Notes in Electrical Engineering, 2012, 2(2):79-84.

[20] BERGH F V, ENGELBRECHT A P. A study of particle swarm optimization particle trajectories[J]. Inf Sci, 2006, 176(8): 937-971.

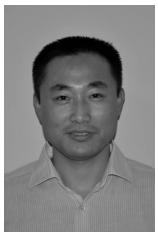
[21] 周殊, 潘炜, 罗斌等. 一种基于粒子群优化方法的改进量子遗传算法及应用[J]. 电子学报, 2006, 34(5):897-901.
ZHOU S, PAN W, LUO B, *et al*. A novel quantum genetic algorithm based on particle swarm optimization method and its application[J]. Acta Electronica Sinica, 2006, 34(5):897-901.

[22] WOLPERT D H, MACREADY W G. No free lunch theorems for optimization[J]. IEEE Trans Evol Comput, 1997, 1(1):67-82.

作者简介：

郭通（1984-），男，江西南昌人，国家数字交换系统工程技术研究中心博士生，主要研究方向为宽带信息网络、高速网络业务管控等。

兰巨龙（1962-），男，河北张北人，博士，国家数字交换系统工程技术研究中心教授、博士生导师，主要研究方向为宽带信息网络、高速路由器核心技术等。



李玉峰（1975-），男，山东烟台人，博士，国家数字交换系统工程技术研究中心讲师，主要研究方向为路由与交换、网络业务识别与控制等。

陈世文（1974-），男，湖南益阳人，国家数字交换系统工程技术研究中心博士生、副教授，主要研究方向为网络异常流量检测等。

（上接第 123 页）

IEEE Journal on Selected in Communications, 2012, 30(10): 1934-1946.

[12] PARSAEEFARD S, SHARAFAT A R. Robust worst-case interference control in underlay cognitive radio networks[J]. IEEE Transactions on Vehicular Technology, 2012, 61(8):3731-3745.

[13] PARSAEEFARD S, SHARAFAT A R. Robust distributed power control in cognitive radio networks[J]. IEEE Transactions on Mobile Computing, 2013, 12(4): 609-620.

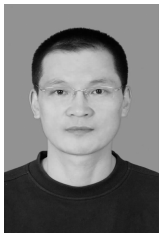
[14] BERTSEKAS D P. Nonlinear Programming 2nd ed[M]. Boston, MA: Athena Scientific, 1999.

[15] PERKINS C E, ROYER E M. Ad-hoc on demand distance vector routing[A]. IEEE WMCSA 1999[C]. New Orleans, 1999. 90-100.

[16] SAVIC V, ZAZO S. Reducing communication overhead for cooperative location using nonparametric belief propagation[J]. IEEE Wireless Communications Letter, 2012, 1(4):308-311.

作者简介：

徐勇军（1986-），男，湖北赤壁人，吉林大学博士生，主要研究方向为认知无线电系统中的资源分配。



赵晓晖（1957-），男，北京人，博士，吉林大学教授、博士生导师，主要研究方向为认知无线电、信号处理理论及在通信中的应用。