文章编号: 1001-0920(2011)02-0201-06

加速收敛的粒子群优化算法

任子晖, 王 坚

(同济大学 CIMS 研究中心，上海 201804)

摘 要: 在基本粒子群优化算法的理论分析的基础上, 提出一种加速收敛的粒子群优化算法, 并从理论上证明了该算法的快速收敛性, 同时对该算法中的参数进行了优化. 为了防止其在快速收敛的同时陷入局部最优, 采用依赖部分最差粒子信息的变异操作. 最后通过与其他几种经典粒子群优化算法的性能比较, 表明了该算法的高效和稳健, 且明显优于现有的几种经典的粒子群算法.

关键词: 粒子群优化；加速收敛；参数优化；变异

中图分类号: TP18 **文献标识码:** A

Accelerate convergence particle swarm optimization algorithm

REN Zi-hui, *WANG Jian*

(CIMS Research Center，Tongji University，Shanghai 201804，China．Correspondent：REN Zi-hui，E-mail: renzihui2006@126.com)

Abstract: An accelerate convergence particle swarm optimization(ACPSO) algorithm is proposed based on analyzing the convergence of basal particle swarm optimization(BPSO) algorithm. The convergence speed of ACPSO algorithm is very quickly through theoretical analysis. Then the parameters in this algorithm are optimized. The mutation operator of depending on segmental worst particles' information is shown to escape the local optimal. The performance of ACPSO algorithm with the optimal parameters is tested on several classical functions by comparing with four classical PSO algorithms. The experimental results show that the ACPSO algorithm is efficient and robust. Especially, the convergence speed of ACPSO is superior to several classical PSO algorithms obviously.

Key words: particle swarm optimization；accelerate convergence；parameters optimization；mutation

1 引 言

粒子群优化(PSO)算法源于对鸟群或鱼群捕食行为的模拟. 自 Kennedy 和 Eberhart[1]于 1995 年提出粒子群优化算法以来, 该算法就得到了广泛的研究和应用. 最具有代表性的几种 PSO 算法的变形有: 文献 [2] 提出的线性递减 PSO 算法 (LDPSO); [3] 给出的带压缩因子的 PSO 算法 (CPSO); [4] 提出的多环拓扑结构的 PSO 算法 (MTPSO). 此外, 还有很多学者对 PSO 算法进行了改进, 如: [5] 提出了一种带有混沌变异算子的量子粒子群优化算法; [6] 给出了一种自适应规模的粒子群优化算法及其应用; [7] 提出了一种自适应的粒子群优化等. 目前, 对 PSO 算法的理论分析研究还不是很完善, 主要有: [8] 在一组假设的前提下, 提出了在这组假设下粒子的位置和速度可近似为一正弦波的形状, 瞬间不同的参数决定了波的频率和振幅; [9] 在 [8] 的基础上作了进一步的研究, 给出了构造因子方法的 3 种模型并给出了仿真结果; [10] 在 [8] 的基础上给出了 PSO 模型的稳定区域和不稳定的区域并对粒子的运动轨迹进行了分析. 此外, 对离散的 PSO 算法[11-14]、多目标的 PSO 算法[15-18]及算法的拓扑结构[4,19-20]等方面的研究都很多. 综上可知, 目前对连续 PSO 算法的改进主要集中在算法参数的各种各样的调整和粒子更新结构的调整上, 目的是使粒子跳出局部最优, 使其全局搜索能力和局部搜索能力达到最佳的平衡状态, 加快收敛速度, 从而提高算法的性能; PSO 的理论研究还很缺乏, 目前主要是对基本 PSO 算法的理论进行了初步的分析研究, 更加深入的分析还有待进一步研究.

收稿日期: 2009-11-21；**修回日期:** 2010-03-27.

基金项目: 国家科技支撑计划项目(2006BAG01A02)；上海市科技发展基金项目(08201201905, 08DZ1120802)；上海市重点学科建设项目(B004).

作者简介: 任子晖(1983–), 女, 博士, 从事计算智能及其应用的研究; 王坚(1961–), 男, 教授, 博士生导师, 从事智能计算、仿真技术等研究.

本文从基本PSO理论出发, 在分析其收敛性的基础上, 提出了一种加速收敛的PSO算法(ACPSO), 并对其进行了简单的理论分析, 以及对ACPSO算法中涉及的参数进行优化. 采用文献[21]中的依赖部分最差粒子信息的变异操作来防止陷入局部最优, 并将优化的ACPSO算法与其他几种经典的PSO算法进行比较分析, 实验结果表明了ACPSO算法的可行性及有效性, 尤其表明了其收敛的快速性.

2 基本的PSO算法及理论分析

PSO算法以模拟鸟的群体智能为特征, 以求解连续变量优化问题为背景. 在PSO算法中, 每只鸟被称之为一个粒子, 每个粒子用其几何位置和速度向量表示, 每个粒子参考自己的既定方向, 所经历的最优方向和整个鸟群所公共认识到的最优方向来确定自己的飞行. 假设在一个D维的目标搜索空间中, 有m个粒子. 其中: 第i个粒子表示为一个D维的向量, $x_i=(x_{i1},x_{i2},\cdots,x_{iD})(i=1,2,\cdots,m)$表示第$i$个粒子在此搜索空间中的位置, $v_i=(v_{i1},v_{i2},\cdots,v_{iD})(i=1,2,\cdots,m)$表示第$i$个粒子的飞翔速度. 设第$i$个粒子迄今为止搜索到的最优位置为$p_i=(p_{i1},p_{i2},\cdots,p_{iD})(i=1,2,\cdots,m)$, 整个粒子群迄今为止搜索到的最优位置为$p_g=(p_{g1},p_{g2},\cdots,p_{gD})(i=1,2,\cdots,m)$. 采用下列公式对粒子群进行操作:

$$v_{i,d}=\omega v_{i,d}+c_1r_1(p_{i,d}-x_{i,d})+c_2r_2(p_{g,d}-x_{i,d}), \tag{1}$$

$$x_{i,d}=x_{i,d}+v_{i,d}. \tag{2}$$

其中: $i=1,2,\cdots,m;d=1,2,\cdots,D$; 学习因子$c_1,c_2$为非负常数; r_1,r_2为[0, 1]间的随机数; $v_{id}=[-v_{\max},v_{\max}],v_{\max}$为常数; ω为惯性系数. 式(1)的第1部分为粒子先前的速度; 第2部分为“认知”部分, 表示粒子本身的思考; 第3部分为“社会”部分, 表示粒子间的信息共享与相互合作.

迭代终止条件根据具体问题一般选为最大迭代次数或粒子群迄今为止搜索到的最优位置满足预定最小适应阀值. 方程(1)和(2)是基本的PSO算法. 仿照文献[22]对BPSO的推导, 将式(1)写成矩阵形式, 得到如下表达式:

$$\begin{bmatrix}v_{t+1}\\x_{t+1}\end{bmatrix}=A_t\begin{bmatrix}v_t\\x_t\end{bmatrix}+\begin{bmatrix}\varphi_1&\varphi_2\\\varphi_1&\varphi_2\end{bmatrix}\begin{bmatrix}P_{i,t}\\p_{g,t}\end{bmatrix}. \tag{3}$$

其中

$$A_t=\begin{bmatrix}\omega&-\varphi\\\omega&1-\varphi\end{bmatrix},\varphi=\varphi_1+\varphi_2=c_1r_1+c_2r_2.$$

令

$$y_t=\begin{bmatrix}v_t\\x_t\end{bmatrix},\Phi_t=\begin{bmatrix}\varphi_1p_{i,t}+\varphi_2p_{g,t}\\\varphi_1p_{i,t}+\varphi_2p_{g,t}\end{bmatrix},$$

于是式(3)转化为

$$y_{t+1}=A_ty_t+\Phi_t. \tag{4}$$

因为r_1,r_2为[0,1]间均匀分布的随机数, 参照文献[3]中的假设, 即$\omega,p_{i,t},p_{g,t}$为时不变的, 且对于种群的最优粒子有$p=p_{i,t}=p_{g,t}$. 因为式(4)中含有随机变量, 直接对其进行研究将会产生较大的误差, 故需要想办法将式(4)中的随机变量变为常量. 参照文献[22]中的假设条件, 当$i\geqslant t$时, 随机变量$v_t,\varphi_1,x_t,\varphi_2$相互独立. 于是当$i\geqslant t$时, 有

$$Ey_{t+1}=\begin{bmatrix}Ev_{t+1}\\Ex_{t+1}\end{bmatrix},\ EA_t=\begin{bmatrix}\omega&-\dfrac{c_1+c_2}{2}\\\omega&1-\dfrac{c_1+c_2}{2}\end{bmatrix},$$

$$E\Phi_t=\begin{bmatrix}\dfrac{c_1p_{i,t}+c_2p_{g,t}}{2}\\\dfrac{c_1p_{i,t}+c_2p_{g,t}}{2}\end{bmatrix}=\begin{bmatrix}\dfrac{c_1p+c_2p}{2}\\\dfrac{c_1p+c_2p}{2}\end{bmatrix}.$$

再设$Q_{t+1}=Ey_{t+1},M=EA_t,Z=E\Phi_t$. 于是式(4)等价于

$$Q_{t+1}=MQ_t+Z. \tag{5}$$

定理1 若BPSO算法的收敛速度与M的特征值有关, 当M特征值的绝对值小于1时, 算法收敛, 且在一定范围内特征值越小收敛越快.

证明 M特征值为

$$\lambda_{1,2}=\frac{1+\omega-\dfrac{c_1+c_2}{2}\pm\sqrt{\Big(-1-\omega+\dfrac{c_1+c_2}{2}\Big)^2-4\omega}}{2}. \tag{6}$$

由矩阵论中的知识可知, 只要满足

$$1-\frac{c_1+c_2}{2}\neq-\omega\pm2\sqrt{\omega}, \tag{7}$$

就存在矩阵A使得下式成立:

$$AMA^{-1}=L=\begin{bmatrix}\lambda_1&0\\0&\lambda_2\end{bmatrix}. \tag{8}$$

如设$AQ_t=H_t$, 用A去乘式(5)的两端可得

$$AQ_{t+1}=AMQ_t+AZ, \tag{9}$$

再根据式(8), 于是式(9)等价于

$$\begin{aligned}&AQ_{t+1}=AA^{-1}LAQ_t+AZ=LAQ_t+AZ\Longrightarrow\\&H_{t+1}=LH_t+AZ=L(LH_{t-1}+AZ)+AZ=\\&L^2H_{t-1}+(L+I)AZ=\\&L^2(LH_{t-2}+AZ)+(L+I)AZ=\\&L^3H_{t-2}+(L^2+L+I)AZ=\cdots=\\&L^{t+1}H_0+(L^t+L_{t-1}+\cdots+L+I)AZ.\end{aligned} \tag{10}$$

由文献[23]中的定理7.1和定理7.3可知, 若需要H_t的迭代收敛, 必须满足L的普半径的绝对值小于1, 且普半径越小, 收敛速度越快, 具体关系为η

$= -\ln\rho(B)$. 其中: η 为收敛速度, $\rho(B)$ 为矩阵 B 的普半径. 矩阵的普半径即为矩阵特征值绝对值的最大值, 即

$$|\lambda_{1,2}| < 1,$$
$$\left|1+\omega-\frac{c_1+c_2}{2} \pm \left(\left(-1-\omega+\frac{c_1+c_2}{2}\right)^2-4\omega\right)^{1/2}\Big/2\right| < 1, \tag{11}$$

且收敛速度与特征值的大小有关. 由于

$$\left(-1-\omega+\frac{c_1+c_2}{2}\right)^2-4\omega \geqslant 0,$$

使得特征值在一定范围内变化, 且特征值越小收敛速度越快. □

3　加速收敛粒子群优化算法及理论分析

根据定理 1 可以改变粒子群优化算法的基本表达式, 使其收敛速度更快. 下面给出一种加速收敛的粒子群优化算法 (ACPSO), 在基本粒子群优化算法的基础上, 使粒子在每次速度迭代过程中偏离速度迭代一个小角度, 而在位置迭代过程中偏离迭代位置一小步, 具体的算法如下:

$$v_{t+1} = [\Theta(\alpha)]^{\beta}[\omega v_t + c_1 r_{1,t}(p_{i,t}-x_t) + c_2 r_{2,t}(p_{g,t}-x_t)], \tag{12}$$

$$x_{t+1} = [\Theta(\alpha)]^{\beta} x_t + v_{t+1}. \tag{13}$$

其中: Θ 表示一个三角函数算子, $[\Theta(\alpha)]$ 的值小于 1, 取 $\Theta = \sin$ 或 $\Theta = \tan$ 或 $\Theta = \cos$; α 为一个角度值, 当 Θ 不同时, α 的取值也是不同的, 对于同一个算子, α 的值不同, 其结果也不相同; β 为一个大于零的常数, 也与算法的性能有很大的关系. 下面将详细讨论 Θ, α, β 的取值与算法性能之间的关系及它们的最优取值或区间. 上述算法与 Clerc 和 Kennedy[3] 提出的带压缩因子的粒子群优化算法 (CPSO) 是不同的, 在本文中, 粒子对自身经历过的位置记忆部分 (惯性部分) 是有选择和继承的, 在对位置的更新策略上是有区别的, 后面将用实验表明本文提出的 ACPSO 算法优于 CPSO 算法.

定理 2　假设矩阵序列 A_i 的普半径为 ρ_A, 矩阵序列 B_i 的普半径为 ρ_B, 那么矩阵序列 A_i 和矩阵序列 B_i 的收敛速度的比值为 $\chi = \ln(\rho_A)/\ln(\rho_B)$.

证明　根据文献 [23] 可知, 矩阵序列 A_i 和矩阵序列 B_i 的收敛速度的比值为

$$\chi = \eta_A/\eta_B = -\ln(\rho_A)/-\ln(\rho_B) = \ln\max|\lambda_A|/\ln\max|\lambda_B|. \tag{14}$$

又因为矩阵的普半径就是矩阵特征值绝对值取最大, 函数 ln 是增函数, 收敛的条件是普半径小于 1, 故收敛速度和特征值的比值类似成正比关系, 即特征值越大, 其收敛速度越慢. □

定理 3　ACPSO 算法的收敛速度大于基本 PSO 算法, 其收敛速度为基本 PSO 算法收敛速度的 $1+\ln(|[\Theta(\alpha)]^{\beta}|)/\ln\max|\lambda_{1,2}|$ 倍.

证明　同理, 采用上述的分析方法可得到 ACPSO 收敛的条件为

$$|\tilde{\lambda}_{1,2}| < 1,$$
$$\left|[\Theta(\alpha)]^{\beta}\left(1+\omega-\frac{c_1+c_2}{2} \pm \left(\left(-1-\omega+\frac{c_1+c_2}{2}\right)^2-4\omega\right)^{\frac{1}{2}}\right)\Big/2\right| < 1. \tag{15}$$

因为

$$\max|\tilde{\lambda}_{1,2}|/\max|\lambda_{1,2}| = |[\Theta(\alpha)]^{\beta}|, \tag{16}$$

故 ACPSO 与基本 PSO 收敛速度的比值为

$$1+\ln(|[\Theta(\alpha)]^{\beta}|)/\ln\max|\lambda_{1,2}|,$$

即 ACPSO 收敛速度为基本 PSO 算法收敛速度的 $1+\ln(|[\Theta(\alpha)]^{\beta}|)/\ln\max|\lambda_{1,2}|$ 倍. 这也说明了 ACPSO 算法的收敛速度大于基本 PSO 算法. □

4　变异操作

文献 [21] 提出当粒子可能连续次找不到最优解时, 则认为粒子的有序性在很大程度上增强了, 这样不利于全局搜索. 从 PSO 的整个系统内的粒子来看, 可能所有的粒子呈现趋同性, 即此时粒子很可能陷入了局部最优. 此时增加粒子的混乱度, 使粒子的进化部分依赖于系统群体的最差信息, 在算法中表现为粒子的速度与群体的最优位置信息及与粒子自身的最优位置信息无关, 而只依赖于群体的最差位置信息. 根据下式来计算 Υ:

$$\Upsilon = \begin{cases} 0, & f(p_g(i)) \leqslant f(p_g(i-1)); \\ \Upsilon+1, & f(p_g(i)) > f(p_g(i-1)). \end{cases} \tag{17}$$

判断 Υ 与 Δ 的关系, 如果 $\Upsilon \geqslant \Delta$, 则按照下式更新粒子的位置:

$$v_{j,d} = \omega v_{j,d} + c_2 r_2(p_{w,d} - x_{j,d}), \tag{18}$$

其中: $p_{w,d}$ 表示粒子的最差位置; 否则, 按式 (1) 的规则更新. 在下面的具体计算中, 取 $\Delta = 15$.

5　数值实验

5.1　参数的选取实验

下面将对 ACPSO 算法中的参数 α, β 及三角函数算子 Θ 进行实验分析, 给出 α, β 的最优值或区间, 并针对三角函数算子 Θ 给出其最优形式. 本文选择了 4 个函数进行数值实验, 分别为: Sphere 函数, Rosenbrock 函数, Griewank 函数和 Rastrigin 函数. 其中: Sphere 函数和 Rosenbrock 函数都是单峰函数, 其余两个函数均为多峰函数. Sphere 函数比较简单, 一般用来测试算法的精度, 考察算法的执行能力; Rosenbrock 函数是非凸的病态函数, 一般用来考察算

法的执行效率; 其他两个函数都有很多的极小点, 一般很难找到算法的全局最优解, 所以常被用来考察算法跳出局部最优解的能力. 下面给出这几种函数的具体形式[1], 如表 1 所示. 其中: D 表示维数; $v_{\max}$ 表示速度的最大限制, 本文粒子的搜索空间均限制在 $[-v_{\max}, v_{\max}]$ 内; f^* 表示函数的最优值.

表 1 测试函数的具体形式及参数

函 数	D	$v_{\max}$	f^*
Sphere $\sum_{i=1}^{n} x_i^2$	30	100	0
Rosenbrock $\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	30	0
Griewank $\frac{1}{4\,000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\frac{x_i}{\sqrt{i}}+1$	30	600	0
Rastrigin $\sum_{i=1}^{D}[x_i^2-10\cos 2\pi x_i+10]$	30	5.12	0

下面首先测试三角函数算子对算法性能的影响, 由定理 3 可知, 当三角函数的值在 (0, 1) 之间时算法是收敛的. 故分别给出当三角函数算子取正弦、余弦和正切时所得最优值的比较, 如图 1 所示. 在实验中, 参数的设置如下: 粒子数取 50, $c_1 = c_2 = 1.454$, $\omega = 0.729$, $\alpha \in [0, \pi/8]$, $\beta = 1$. 由于取这样的参数值能满足式 (11) 和 (14), 使得基本的 PSO 算法和 ACPSO 算法都满足收敛条件. 测试计算机的性能为: Intel(R) Celeron(R) M processor 1.50GHz 1.50GHz, 704MB PC; 测试软件为 Matlab7.0.

由图 1 可以看出, 当三角函数算子 $\Theta = \sin$ 或 $\Theta = \tan$ 时的效果优于 $\Theta = \cos$, 算法 ACPSO 中的三角函数算子的最优取值应该为 $\Theta = \sin$, 此时算法收敛速度最快, 算法的性能最优.

下面研究 α 的取值区间对算法性能的影响, 在测试时 $\Theta = \sin$. 参数的设置同上. 经过初步的实验可知, α 的下界取为大于零的任意小值. 在此给出几种区间选择, 分别为: $\alpha \in [0, \pi/8]$, $\alpha \in [0, \pi/6]$, $\alpha \in [0, \pi/4]$ (其他区间可类似转化为此 4 种情况). 具体的实验结果如图 2 所示.

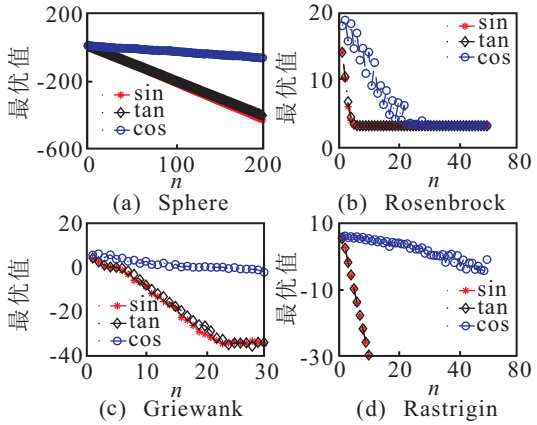

图 1 当三角函数不同时求解函数的对比

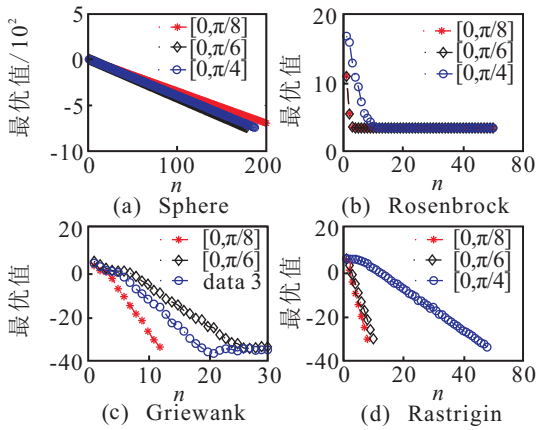

图 2 当 α 不同时求解函数的对比

从图 2 可以看出, 当 $\alpha \in [0, \pi/8]$ 时, 函数值收敛得更快. 对于不同的函数, α 值的范围对其函数值收敛速度的影响也是不同的. 如对于单峰的 Sphere 函数和 Rosenbrock 函数而言, 当 $\alpha \in [0, \pi/8]$ 和 $\alpha \in [0, \pi/6]$ 时函数值的收敛效果几乎是一致的; 而对于剩下的两个比较复杂的多峰函数而言, $\alpha \in [0, \pi/6]$ 的结果并不理想; 总而言之, 当 $\alpha \in [0, \pi/8]$ 区间时的平均效果最优, 因此这里的 $\alpha \in [0, \pi/8]$.

下面对参数 β 进行实验分析, 在实验中 $\Theta = \sin$, $\alpha \in [0, \pi/8]$. 其余参数设置同上. 由于当 β 的取值大于 1 时, 收敛速度很快. 为了能更加看清楚 β 的取值对算法性能的影响, 在下面的测试中最大迭代次数都取值较小. 具体结果如图 3 所示.

由图 3 可以看出, β 的值越大, 函数值收敛的越快. 但是总体来看, 当 $\beta = 3$ 时, 函数值的收敛速度和 $\beta \geqslant 4$ 时函数值的收敛速度几乎是相同的, 故可认为当 $\beta = 3$ 时的 ACPSO 的性能和 $\beta \geqslant 4$ 时 ACPSO 的性能相差是很小的, 即可认为当 $\beta = 3$ 时 ACPSO 的性能已达到最优. 因此取 $\beta = 3$. 综合以上 3 类实验, 可以给出 ACPSO 的最优形式如下:

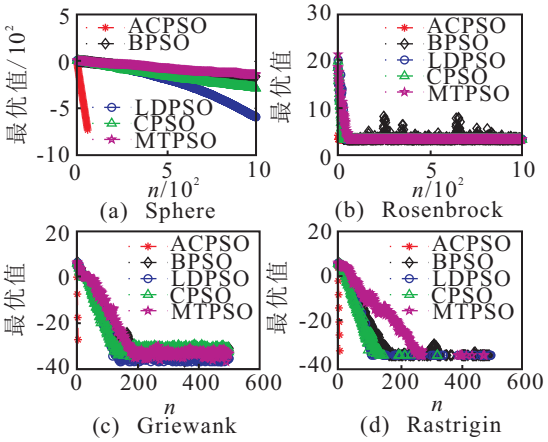

图 3 当 3 个 β 不同时求解函数的对比

$$v_{t+1} = [\sin(\alpha)]^3[\omega v_t + c_1 r_{1,t}(p_{i,t} - x_t) + c_2 r_{2,t}(p_{g,t} - x_t)], \tag{19}$$

$$x_{t+1} = [\sin(\alpha)]^3 x_t + v_{t+1}, \tag{20}$$

$$\alpha \in [0, \pi/8]. \tag{21}$$

5.2　性能测试实验及分析

将最优的ACPSO算法性能与几种具有代表性的经典PSO算法的性能进行对比, 在此选取基本的PSO算法(BPSO)[1], 线性递减的PSO算法(LDPSO)[2], 带压缩因子的PSO算法(CPSO)[3]及多环拓扑结构的PSO算法(MTPSO)[4]. 这几种算法中的参数均采用经典的参数, 即: BPSO算法中参数的设置为 $c_1 = c_2 = 1.454, \omega = 0.729$; LDPSO算法中参数的设置为 $c_1 = c_2 = 1.454, \omega = 0.729, \omega_{\min} = 0.2$; CPSO算法中参数的设置为

$$\chi = \frac{2}{|2 - \rho - \sqrt{\rho^2 - 4\rho}|},\ \rho = 4.1;$$

MTPSO算法中参数的设置为 [0, 4.1] 间的随机数. 在MTPSO中, 当粒子不处于边缘时均采用多环拓扑结构; 当粒子处于边缘时, 采用BPSO算法的拓扑结构; 相应的参数也采用BPSO算法中的参数. 具体结果如图4所示.

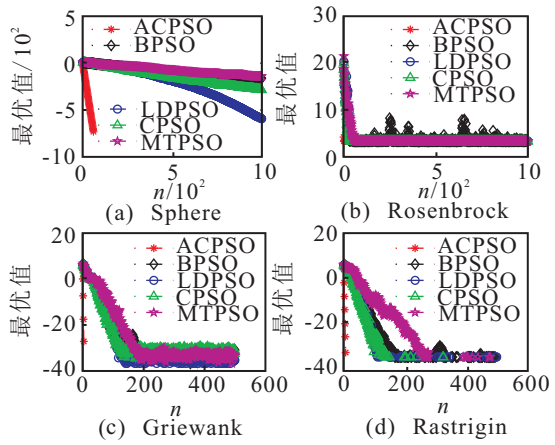

图 4　求解 Sphere 函数的对比 (NC = 1 000)

从图4可以看出, 对于Sphere函数而言, ACPSO算法明显优于其他几种方法, 其函数值的拉格朗日值是呈直线下降的, 这就意味着使用ACPSO求解Sphere函数时函数值收敛得非常快, 其次是LDPSO算法, 而BPSO算法的性能最差. 从求解Rosenbrock函数的结果能够看出, 利用ACPSO算法求解Rosenbrock函数, 在迭代的初期函数值收敛也很快, 后期则和其他算法一样陷入了局部最优, 在迭代的初期其他几种算法的函数值也是呈下降趋势的, 但在迭代的过程中BPSO算法一直处于震荡状态. 采用ACPSO算法求解Griewank函数和Rastrigin函数时函数值的收敛速度是非常快的, 从图4能够看到, 其函数值的拉格朗日值在迭代进行很短的时间内即呈直线下降状态, 而其他4种算法最后得到的结果相差不大, 但收敛速度明显劣于ACPSO算法, 其中MTPSO算法求解Griewank函数和Rastrigin函数的效果最差.

综上所述, ACPSO算法的收敛速度明显快于BPSO算法, LDPSO算法, CPSO算法和MTPSO算法. 这与定理3的推导结果一致, 但从图4可以看出, 该算法的缺点是仍可能陷入局部最优. 通过对文中4个经典函数(包括单峰函数和多峰函数)的测试结果可知, 采用ACPSO算法求解最后得到的最优值都是有效的, 且通过ACPSO的表达式(12)和(13)也可以看出, 在粒子每次迭代时, $|[\Theta(\alpha)]^\beta|$ 都是变化的, 在每次迭代时, $|[\Theta(\alpha)]^\beta|$ 的值都是不同的. 这说明粒子的多样性很强, 进一步说明了算法早熟收敛出现的概率是很小的. 同时, 文中也给出了防止早熟收敛而采用变异的策略. 下一步工作是, 如何防止粒子陷入局部最优或对陷入局部最优的粒子采用合适的变异操作来进一步提高算法的性能.

6　结　论

本文依据基本粒子群优化(BPSO)算法的理论基础, 提出一种加速收敛的粒子群优化(ACPSO)算法, 并对该算法的理论进行了分析研究. 从理论上证明了算法的收敛速度明显优于BPSO算法, 并对其中涉及的参数进行了实验分析后选取了最优的参数. 通过对几个典型函数的性能实验分析, 表明了本文算法的可行性及有效性, 并表明了其收敛速度快的特性.

参考文献(References)

[1] Kennedy J, Eberhert R. Particle swarm optimization[C]. IEEE Int Conf on Neural Networks. Perth, 1995: 1942-1948.

[2] Shi Y, Eberhert R. Empirical study of particle swarm optimization[C]. Int Conf on Evolutionary Computation. Washington: IEEE, 1999: 1945-1950.

[3] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multi-dimensional complex space[J]. IEEE Trans on Evolutionary Computation, 2002, 6(1): 58-73.

[4] Carmelo J, Bastos-Filho A, Marcel P Caraciolo, et al. Multi-ring dispersed particle swarm optimization[C]. Eighth Int Conf on Hybrid Intelligent Systems. Barcelona, 2008: 25-30.

[5] Leandro dos Santos Coelho. A quantum particle swarm optimizer with chaotic mutation operator[J]. Chaos, Solitons and Fractals, 2008, 37(5): 1409-1418.

[6] Chen D B, Zhao C X. Particle swarm optimization with adaptive population size and its application[J]. Applied Soft Computing, 2009, 9(1): 39-48.

[7] Jin Yisu, Joshua Knowles, Lu Hongmei, et al. The landscape adaptive particle swarm optimizer[J]. Applied Soft Computing, 2008, 8(1): 295-304.

[8] Ozcan E, Mohan C K. Particle swarm optimization: surfing the waves[C]. Proc of Congress on Evolutionary Computation. Washington: IEEE, 1999: 1939-1944.

[9] Clerc M, Kennedy J. The particle swarm-explosion stability and convergence in a multi-dimensional complex space[J]. IEEE Trans on Evolutionary Computation, 2002, 6(1): 58-73.

[10] Trelea I C. The particle swarm optimization algorithm: Convergence analysis and parameter selection[J]. Information Processing Letters, 2003, 85(6): 317-325.

[11] Tseng Chao-Tang, Liao Ching-Jong. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem[J]. European J of Operational Research, 2008, 191(2): 360-373.

[12] Ali Husseinzadeh Kashan, Behrooz Karimi. A discrete particle swarm optimization algorithm for scheduling parallel machines[J]. Computers and Industrial Engineering, 2009, 56(1): 216-22.

[13] Pan Quan-Ke, Fatih Tasgetiren M, Liang Yun-Chia. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem[J]. Computers and Operations Research, 2008, 35(9) : 2807-2839.

[14] Davide Anghinolfi, Massimo Paolucci. A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times[J]. European J of Operational Research, 2009, 193(1): 73-85.

[15] Leong Wen-Fung, Yen Gary G. PSO-based multiobjective optimization with dynamic population size and adaptive local archives[J]. IEEE Trans on Systems, Man, and Cybernetics, Part B: Cybernetics, 2008, 38(5): 1270-1293.

[16] Brits R, Engelbrecht A P, F van den Bergh. Locating multiple optimization using particle swarm optimization[J]. Applied Mathematics and Computation, 2007, 189(2): 1859-1883.

[17] Praveen Kumar Tripathi, Sanghamitra Bandyopadhyay, Sankar Kumar Pal. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients[J]. Information Sciences, 2007, 177(22): 5033-5049.

[18] Yang Junjie, Zhou Jianzhong, Liu Li, et al. A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization(MOPSO)[J]. Computers and Mathematics with Applications, 2009, 57(11/12): 1995-2000.

[19] James Kennedy, Rui Mendes. Neighborhood topologies in fully informed and best-of-neighborhood particle swarms[J]. IEEE Trans on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2006, 36(4): 515-519.

[20] Takeshi Korenaga, Nobuhiko Kondo, Toshiharu Hatanaka, et al. Topology-based personal selection in multi-objective particle swarm optimization[C]. SICE Annual Conf. Tokyo, 2008: 3465-3469.

[21] 任子晖, 王坚. 基于改进熵的粒子群优化算法[J]. 系统工程, 2009, 27(8): 106-113.
(Ren Z H, Wang J. Improved particle swarm optimization algorithm based on entropy[J]. System Engineering, 2009, 27(8): 106-113.)

[22] 金欣磊, 马龙华, 吴铁军, 等. 基于随机过程的PSO收敛性分析[J]. 自动化学报, 2007, 33(12): 1263-1268.
(Jin X L, Ma L H, Wu T J, et al. Convergence analysis of the particle swarm optimization based on stochastic processes[J]. Acta Automatica Sinica, 2007, 33(12): 1263-1268.)

[23] 聂铁军. 数值计算方法[J]. 西安: 西北工业大学出版社, 1990.
(Nie T J. Numerical calculation methods[J]. Xi'an: Northwestern Polytechnical University Press, 1990.)

(上接第200页)

[12] Leong B, Mitra S, Liskov B. Path vector face routing: Geographic routing with local face information[C]. Proc of the 13th IEEE Int Conf on Network Protocols. Washington DC: IEEE Press, 2005: 147-158.

[13] 贺鹏, 李建东. 基于Delaunay三角剖分的Ad Hoc网络路由算法[J]. 软件学报, 2006, 17(5): 1149-1156.
(He P, Li J D. A routing algorithm for ad hoc net-works based on delaunay[J]. J of Software, 2006, 17(5): 1149-1156.)

[14] Fang Q, Gao J, Guibas L. Locating and bypassing routing holes in sensor networks[C]. Proc of the 23rd IEEE Communications Society. Hong Kong: IEEE Press, 2004: 2458-2468.

[15] Chou C H, Ssu K F, Jiau H C. Geographic forwarding with dead-end reduction in mobile ad hoc networks[J]. IEEE Trans on Vehicular Technology, 2005, 57(4): 2375-2386.

[16] Zou L, Lu M, Xiong Z. A distributed algorithm for the dead end problem of location based routing in sensor networks[J]. IEEE Trans on Vehicular Technology, 2005, 54(4): 1509-1522.

[17] N Arad, Shavitt Y. Minimizing recovery state in geographic ad-hoc routing[J]. IEEE Trans on Mobile Computing, 2009, 8(2): 203-217.