融合随机逼近算法的粒子群优化算法①

罗金炎

(闽江学院 数学系, 福州 350108)

摘 要: 针对粒子群优化算法(PSO)在寻优进程中的缺陷, 提出一种融合随机逼近算法的粒子群优化算法, 该算法选择合适时机将随机逼近算法融入粒子群优化算法维持种群的多样性, 并且在算法寻优进程中充分利用已有的计算资源提高算法寻优效率, 最后通过典型标准函数数值实验表明, 改进后的粒子群优化算法寻优速度快、精度高、具较好的稳定性.

关键词: 粒子群优化算法; 随机逼近; 全局最优位置; 函数优化

Improved Particle Swarm Optimizer with Stochastic Approximation

LUO Jin-Yan

(Department of Mathematics, Minjiang University, Fuzhou 350108, China)

Abstract: In order to overcome the shortcomings of the particle swarm optimization(PSO), an improved particle swarm optimization based on simultaneous perturbation stochastic approximation(SPSA) method is proposed. It embeds SPSA into PSO as a local search operator in the proper time, and makes use of the computing resources available in the optimization process. Numerical experiments for benchmark functions have been done, The results indicate that the proposed algorithm performs better than the existing ones in terms of efficiency, accuracy and stability.

Key words: particle swarm optimization; stochastic approximation; global best position; function optimization

1 引言

通过对生物群体的观察和研究发现, 生物群体内个体间的合作与竞争等复杂行为产生的群体智能, 往往能为某些特定的问题提供高效的解决方法. Kennedy 等受鸟群觅食行为的启发,于 1995 年提出粒子群优化算法(PSO)[1], 之后, Shi 和 Eberhart 在原有基础上提出了惯性权重的概念[2], 并对基本算法中的粒子速度更新公式进行了修正, 以获得更佳的优化效果, 该式一般作为 PSO 算法的标准形式.与进化算法相比, PSO 算法保留了基于种群的全局搜索策略, 但其所采用的速度—位移搜索模型操作简单, 避免了复杂的进化操作, 在解决复杂优化等许多的实际问题中取得了许多成功的应用[3].

作为随机搜索算法, 粒子群算法同其他的进化算法一样在寻优过程中存在一些缺陷. 首先同多数的随机优化算法类似, 粒子群算法的粒子运动轨迹依赖于随机参数的取值, 所以随机参数的取值将影响整个粒子群的搜索能力[4]. 其次指导群体粒子搜索方向的当前的全局最优位置粒子(gbest)与其他粒子间的信息传递方式, 容易造成缺乏多样性的相似粒子, 如此增加了算法收敛于局部最优的可能, 这是导致算法在多维多峰搜索空间里“早熟”的主要原因. 因为, 在粒子群算法的运行过程中, 当前时刻全局最优位置粒子(gbest)是最关键的主角, 然而其演化公式极为低级, 粒子一旦成为 gbest 就只能滞留在个体最优位置, 算法的社会和个体成分因素均对其速度的演化基本失效.

针对算法缺陷, 近几年来许多专家学者提出了改进算法[5-7], 但这些改进大多着眼于 PSO 算法的参数选择或动态修改搜索策略, 难以克服 PSO 算法易陷入局部极小的固有弱点.鉴于上述分析, 如何升级全局最优

① 基金项目:国家自然科学基金(11301255);福建省中青年教师教育科研项目(JK2013040)
收稿时间:2014-10-14;收到修改稿时间:2014-12-08

粒子位置gbest的迭代公式显得尤为重要. 有人设法将粒子群算法融合其他随机优化算法来改进, 克服算法缺陷, 特别是将随机逼近算法融合到粒子群算法中, Maeda 和 Kuratani 将同步扰动随机算法[8]的梯度估计值直接嵌入到粒子群算法的速度演化公式中[9], 来提高粒子群算法的寻优能力和多样性, 虽然他们提出了四种选择采用这种混合演化公式的粒子群的不同模式, 但是此法缺乏针对性选择需要改进的粒子, 造成计算资源的浪费; 而 Serkan Kiranyaz,Turker Ince 和 Moncef Gabbouj 等则将每次的全局最优粒子位置运用同步扰动随机算法进行更新[10], 是仅对当前单独的全局最优粒子进行更新, 如此, 可能因为没充分利用到算法所搜索到的众多最优粒子, 会错过真正的全局最优粒子位置, 同样也在某种程度上造成计算成本的提高. 本文也将随机逼近算法融入粒子群算法中, 但在算法中引用一个阈值作为利用随机逼近算法来更新全局最优粒子位置的评判依据, 并且在随机逼近算法的进化中运用粒子群算法已经搜索到的最优粒子群位置作为初始值进行迭代, 如此, 更具针对性, 也充分利用已有的计算资源, 提高算法的效率. 最后在数值实验中, 证明了算法的效果.

2 标准的粒子群优化算法[2]

设第 i 个粒子位置为 $x_i=(x_{i1},x_{i2},...,x_{in})^T$, 其位移为 $v_i=(v_{i1},v_{i2},...,v_{in})^T$. 它迄今搜索到的个体极值为 $p_i=(p_{i1},p_{i2},...,p_{in})^T$, 而迄今搜索到的种群极值为 $p_g=(p_{g1},p_{g2},...,p_{gn})^T$.每一次迭代, 粒子通过两个极值 p_i 和 p_g 来更新其速度和位置.粒子在解空间内不断跟踪个体极值和邻域极值进行搜索, 直到满足迭代停止条件, 即达到规定的迭代次数或满足规定的误差标准.

按追随当前已知的最优位置的原理, 粒子 x_i 将按式(1)改变位移方向和步长.

$$\begin{cases} v_{id}(t+1)=wv_{id}(t)+c_1r_1(p_{id}(t)-x_{id}(t))+c_2r_2(p_{gd}(t)-x_{id}(t)) \\ x_{id}(t+1)=x_{id}(t)+v_{id}(t+1) \end{cases} \quad (1)$$

其中, $d=1,2,...,n$, n 为解空间的维数, 即自变量个数, $i=1,2,...,m$, m 为种群规模, t 为进化代数, r_1 和 r_2 为分布与 $[0,1]$ 之间的随机数, c_1 和 c_2 为位移变化的限定因子, w 为惯性权重.

3 基于随机逼近的粒子群优化算法

3.1 随机逼近算法

同步扰动随机逼近算法(SPSA)是 Spall 在 1987 年根据 Kiefer-Wolforwitz(KW)随机逼近算法改进而成[11]. KW 算法是以有限差分梯度逼近为基础, 它通过估计目标函数的梯度信息来逐渐逼近最优解, 在每次梯度逼近中需要利用目标函数的 2p 个估计值(p 为向量的维数); 而 SPSA 算法在每次梯度逼近中只利用了目标函数的两个估计值, 与向量的维数无关, 从而大大减少了用于估计梯度信息的目标函数的测量次数, 这在解决高维问题以及多变量问题的优化中, 具有其独特的优越性. 若以 $\hat{\theta}_k$ 表示 θ 在第 k 次迭代的变量值, SPSA 算法的迭代公式为:

$$\hat{\theta}_{k+1}=\hat{\theta}_k-a_kg(\hat{\theta}_k) \quad (2)$$

$g(\hat{\theta}_k)$ 表示目标函数 $L(\theta)$ 的梯度估计值, 其公式为:

$$g(\hat{\theta}_k)=\frac{L(\hat{\theta}_k+c_k\Delta_k)-L(\hat{\theta}_k-c_k\Delta_k)}{2c_k\Delta_k} \quad (3)$$

其中, 增益序列 a_k 的一般形式为 $a_k=a/(k+A)^\alpha$, 参数 $c_k=c/k^\gamma$ (a,c,α,γ为正数, $A\geq 0$). SPSA 算法的执行步骤为:

步骤 1: $k=1$, 并初始化变量估计值;

步骤 2: 产生增益序列 $\{a_k\}$, 参数列 $\{c_k\}$ 及独立同分布且均值为零的 p 维向量 Δ_k;

步骤 3: 产生目标函数的两个估计值 $L(\hat{\theta}_k+c_k\Delta_k)$ 和 $L(\hat{\theta}_k-c_k\Delta_k)$;

步骤 4: 依据(3)式产生梯度估计值;

步骤 5: 依据(2)式更新变量 $\hat{\theta}_k$;

步骤 6: 若达到迭代总次数, 则结束; 否则 $k=k+1$, 转步骤 2.

步骤 2 中, α 和 γ 可以取其渐近最优值(分别为 1 和 $1/6$), c 一般取噪声的标准偏差, A 根据具体情况取值, 一般不超过期望迭代最大数的十分之一[11].

3.2 基于随机逼近的粒子群优化算法

根据 SPSA 算法的特点, 将 SPSA 算法融合到 PSO 算法中, SPSA 算法具有较强的局部挖掘能力, PSO 算法具有较强的全局开拓能力. 当 PSO 算法陷入早熟停滞时调用 SPSA 算法进行深度挖掘. 调用 SPSA 算法仅用来修正 gbest 粒子, 这就需要将 SPSA 算法适当地嵌入到 PSO 算法代码中, 这种嵌入方式不会改变 PSO 的

内部结构而只是影响 gbest 粒子的运动.

为节约计算资源、提高算法效率, 我们设置一个域值作为是否调用 SPSA 算法的依据, 并且在调用 SPSA 算法中, 利用前期 PSO 算法搜索到的 gbest 粒子位置作为 SPSA 算法的初始变量估计值; gbest 粒子位置的间距值作为 SPSA 算法梯度估计值公式中差分步长 c 的选值依据, 选取全局最优粒子群位置平均间距作为差分步长 c 的选值.

阈值的设置借用文献[12]判别早熟停滞的方法, 在粒子群迭代过程中, 当得到的最优解在连续次数的迭代过程中都无变化, 便可认为算法有停滞的可能, 表明粒子群已经或者即将限于局部最优解, 阈值设置为最大迭代次数(MAX). MAX 的值可根据求解的问题规模预先指定, 也可根据实验的结果选取适合特定函数优化的值. MAX 的取值越大, 表明判断早熟停滞的标准越宽松. 用一个计数器记录到目前为止停滞的代数 SG, 只要得到的最优解与上次相比不变时, 就将其加 1, 否则将其清 0. 当 SG 达到上限 MAX 时, 说明算法可能停滞, 此时, 有必要调用随机逼近 SPSA 算法对目前的全局最优解进行升级调整来改变粒子的运行轨迹, 使其跳出局部最优解.

算法执行步骤为:

步骤 1: 随机初始化粒子群各粒子的位置和速度, 计算各粒子的适应值 f_i, 令 $p_i = x_i$, p_g 为最优 f_i 对应的 x_i.

步骤 2: 根据(1)式更新粒子速度、位置, 计算种群中各粒子的适应值, 依据适应值更新粒子的个体最优位置 p_i, 设 X_g 是本次迭代中所有粒子找到的最优位置, p_g 为到目前为止所有粒子得到的全局最优位置, 若 $f(X_g) < f(p_g)$, 则令 $p_g = X_g$, SG=0, 并记录各个全局最优粒子位置间的平均间距 d, 否则 SG 加 1. 如果 SG=MAX, 则令 $k=1$, $\hat{\theta}_1 = p_g$, $c = d$, 执行步骤 3; 否则执行步骤 4.

步骤 3: 计算估计值 $L(\hat{\theta}_k + c_k\Delta_k) = f(\hat{\theta}_k + c_k\Delta_k)$ 和 $L(\hat{\theta}_k - c_k\Delta_k) = f(\hat{\theta}_k - c_k\Delta_k)$, 依据(3)式产生梯度估计值 $g(\hat{\theta}_k)$, $\hat{\theta}_{k+1} = \hat{\theta}_k - a_k g(\hat{\theta}_k)$, $k = k+1$, 如果 $k < \lambda \cdot MAX$, 返回步骤 3, 否则, 令 $p_g = \hat{\theta}_k$, SG=0, 执行步骤 4.

步骤 4: $j = j+1$, 如果 $j < M$, 则返回步骤 2, 否则, 迭代结束, 输出 p_g 及其对应的 $f(p_g)$ 即为最优值.

4 数值实验分析

为了测试算法解决优化问题的可行性和效果, 利用表 1 中所列 8 个测试函数对该算法进行测试, 并同标准 PSO 算法和文献[9,10]改进的 PSO 算法进行比较. 为公平各算法的参数设置均一致, 其中 $c_1 = c_2 = 2.0$, 惯性权重 w 采用由 1.0 到 0.4 线性递减, 粒子群 m 取 30, 达优准则为目标函数值小于 $\varepsilon = 0.001$, 最大迭代代数 M=1000, 试验次数均为 100, 设置停滞判定代数 MAX=10.

测试结果如表 2 所示, 从表中可得, 达优率方面, 三种融合了随机逼近算法的 PSO 算法不相上下, 本文算法略好于另外两算法, 但明显比标准 PSO 算法好; 在函数测试次数上, 本文算法明显好于其他算法, 而文献[9,10]的算法并不优于标准 PSO 算法, 因为在此两算法的进程中每次迭代都调用了 SPSA 过程而增加了目标函数计算次数; 函数适应值均方差方面, 本文算法好于其他算法, 精确度较高.

为观察比较四种算法的达优率、适应值函数计算次数及函数适应值均方差的整体效果, 引用文献[13]运用的性能指标(PI)来评价, 各算法性能指标的公式如下:

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}(k_1\alpha^i{}_1 + k_2\alpha^i{}_2 + k_3\alpha^i{}_3),$$

其中, $\alpha^i{}_1 = \dfrac{Sr^i}{Tr^i}$, $\alpha^i{}_2 = \begin{cases} \dfrac{Mf^i}{Af^i}, if \quad Sr^i > 0 \\ 0, \quad if \quad Sr^i = 0 \end{cases}$,

$$\alpha^i{}_3 = \begin{cases} \dfrac{Mo^i}{Ao^i}, if \quad Sr^i > 0 \\ 0, \quad if \quad Sr^i = 0 \end{cases}, \quad i = 1,2,...,N_p.$$

表 1 测试函数

序号	函数名	表达式	搜索区间	最优值
1	Sphere	$Min \quad f(x)=\sum_{i=1}^{n} x_i^2$	$-100\le x_i\le 100$	0
2	Rosenbrock	$Min \quad f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$-30\le x_i\le 30$	0
3	Rastrigin	$Min \quad f(x)=10n+\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)]$	$-5.12\le x_i\le 5.12$	0
4	Griewank	$Min \quad f(x)=1+\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	$-600\le x_i\le 600$	0
5	CosineMixture	$Min \quad f(x)=0.1n-0.1\sum_{i=1}^{n}\cos(5\pi x_i)+\sum_{i=1}^{n} x_i^2$	$-1\le x_i\le 1$	0
6	Exponential	$Min \quad f(x)=1-\exp(-0.5\sum_{i=1}^{n} x_i^2)$	$-1\le x_i\le 1$	0
7	Zakharov's	$Min \quad f(x)=\sum_{i=1}^{n} x_i^2+[\sum_{i=1}^{n}(0.5i)x_i]^2+[\sum_{i=1}^{n}(0.5i)x_i]^4$	$-5.12\le x_i\le 5.12$	0
8	Cigar	$Min \quad f(x)=x_i^2+100000\sum_{i=2}^{n} x_i^2$	$-10\le x_i\le 10$	0

表 2 各算法的计算结果比较

指标	算法	*F1*	*F2*	*F3*	*F4*	*F5*	*F6*	*F7*	*F8*
达优率 SR	标准 PSO	100	0	0	37	73	52	0	73
	PSO[9]	100	16	21	59	81	83	6	100
	PSO[10]	100	17	27	67	84	82	8	100
	本文算法	100	18	28	81	88	95	10	100
函数测试平均次数 AFE	标准 PSO	24588	50000	50000	39216	39944	39972	50000	31166
	PSO[9]	20510	50000	44930	39246	39714	39526	50000	35884
	PSO[10]	20089	50000	45312	38916	39661	39330	50000	32989
	本文算法	16804	47638	40236	28558	30642	20150	38123	20204
函数适应值均方差 AE	标准 PSO	0.000902	232.87	70.01238	0.000693	0.2568	0.6443	26.2914	1.46012
	PSO[9]	0.000703	170.6	41.00109	0.000398	0.01656	0.00463	1.6431	0.00961
	PSO[10]	0.000683	151.3	32.602	0.0001566	0.02131	0.00487	1.5967	0.00997
	本文算法	0.000533	98.45	19.412	0.000089	0.001088	0.00185	1.5144	0.00103

Sr^i 表求解第 i 个问题的达优次数, Tr^i 表求解第 i 个问题的测试总次数; Mf^i 表求解第 i 个问题时所有算法进程中适应值函数平均计算次数的最小值, Af^i 表当前算法求解第 i 个问题时适应值函数的平均计算次数; Mo^i 表求解第 i 个问题时所有算法中最小的目标函数均值, Ao^i 表当前算法求解第 i 个问题的目标函数均值; N_p 表示测试问题的个数. k_1、k_2 和 k_3 ($0\le k_1,k_2,k_3\le 1$ 且 $k_1+k_2+k_3=1$)分别表示对应达优率、适应值函数计算次数及目标函数均值各个指标分量的权重. 性能指标 PI 是 k_1、k_2 和 k_3 的三元函数. 由于在三维图中比较各算法的性能指标更为困难而且三变量满足 $k_1+k_2+k_3=1$, 我们可以设置其中两变量取相等值, 这样 PI 就成为单变量函数. 取值模式如下:

(i) $k_1=w, k_2=k_3=\frac{1-w}{2}, 0\le w\le 1$;

(ii) $k_2=w, k_1=k_3=\frac{1-w}{2}, 0\le w\le 1$;

(iii) $k_3=w, k_1=k_2=\frac{1-w}{2}, 0\le w\le 1$.

数值实验相对应于三种模式的性能指标结果分别如图 1、2、3 所示, 在图中横坐标表权重 w, 纵坐标表性能指标 PI. 在模式 1 中, 适应值函数平均计算次数和目标函数均值取相等的权重, 达优率权重变化, 图 1 显示了四种算法的性能指标值, 本文算法的性能指标值明显高于其他三种算法, 说明本文算法的达优率明显高; 模式 2 中, 达优率和目标函数均值取相等的权重, 函数计算次数权重变化, 从图 2 中可以看出, 本文算法在此模式的性能指标好于其他算法, 标准的 PSO 算法在权重 w 取值比较大时性能指标好于文献

[6, 7]的算法, 说明后两种算法在计算效率没有优势; 模式 3 中, 达优率和适应值函数平均计算次数取相等的权重, 目标函数均值权重变化, 从图 3 中可以看出, 结果和图 1 相似, 本文算法得到最优解的精确度方面好于其他算法.

为了直观地反映本文算法的收敛和优化性能, 本文给出了各算法对测试函数的迭代曲线图, 本文选择其中四种函数($F1$、$F2$、$F4$、$F7$), 单峰和多峰各两种, 如图 4 所示. 由图可得, 本文算法比其他算法的收敛速度要快, 目标函数值也能得到更好的水平. 随着迭代进行, 标准 PSO 算法容易陷入早熟收敛使得适应值停滞, 进化停止. 其它三种算法在避免早熟提高寻优精度方面具有相似结果, 不过本文算法收敛速度优于另外两种算法, 这是因为在进化进程中设置阈值作为调用 SPSA 算法的依据避免了重复不必要的调用, 节约了计算资源、提高了算法效率.

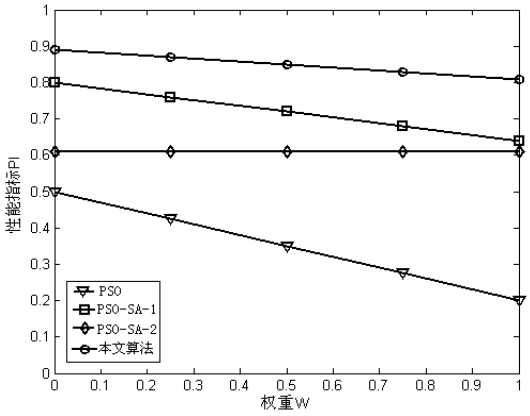


图 1　模式 1 的性能指标图

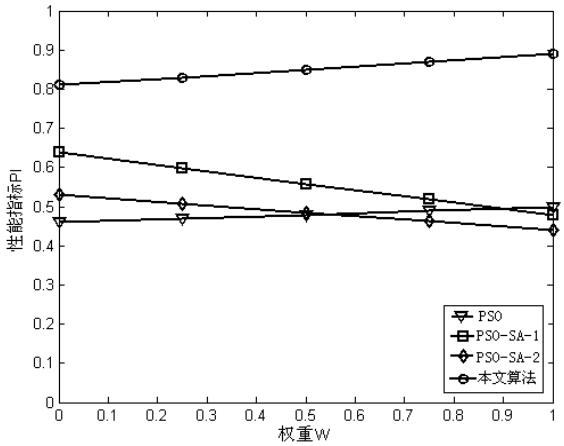


图 2　模式 2 的性能指标图

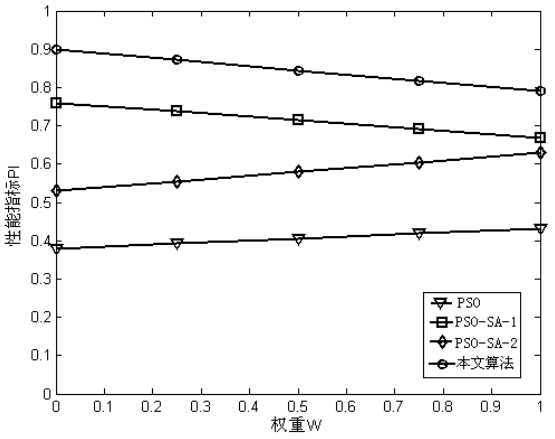
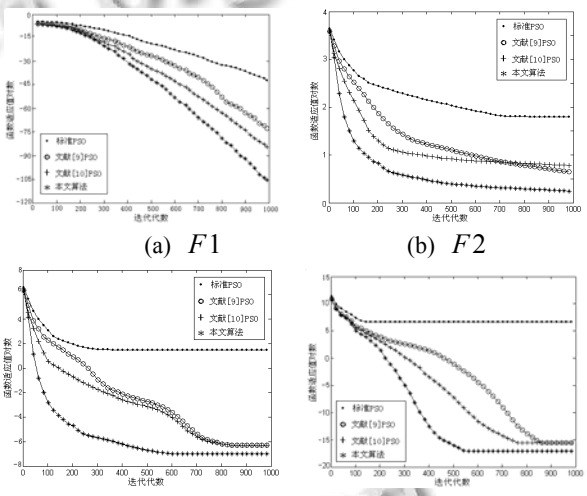


图 3　模式 3 的性能指标图



(a) $F1$　(b) $F2$

(c) $F4$　(d) $F7$

图 4　各算法在不同函数上的适应度进化曲线图

从实验数值结果和各模式的性能指标变化图和收敛迭代图, 可以得出本文算法在优化效率和稳定性方面明显好于其他算法.

5　结语

本文注重对粒子群优化算法中全局最优粒子的更新模式, 将随机逼近算法融入粒子群算法中以增加多样性避免早熟, 为提高计算效率避免计算资源的浪费, 在算法中设置阈值来研判调用随机逼近算法的时机, 并且利用前期粒子群算法得到的 gbest 粒子位置间距的平均值来作为随机逼近算法步长的取值, 从而有效地减少无效迭代, 大大提高算法的收敛速度和优化结果的精度. 减小了在进化过程中停滞于局部最优解的概率, 提高了搜索效率.

参考文献

1 Kennedy J, Eberhart RC. Particle swarm optimization. Proc. of IEEE International Conference on Neural Networks. Perth. Australia. 1995. 1942–1948.

2 Shi YH, Eberhart RC. A modified particle swarm optimizer. Proc. of the IEEE International Conference on Evolutionary Computation. Piscataway, NJ, Anchorage, AK USA. IEEE Service Center. 1998. 69–73.

3 Poli R, Kennedy J, Blackwell T. Particle swarm optimization. Swarm Intelligence, 2007, 1: 33–57.

4 Kennedy J. The particle swarm: Social adaptation of knowledge. Proc. of 1997 International Conference on Evolutionary Computation. Indianapolis. 1997. 303–308.

5 卢峰,高立群.基于改进粒子群算法的优化策略.东北大学学报(自然科学版),2011,32(9):1221–1224.

6 姚灿中,杨建梅.基于变惯性权重及动态邻域的改进 PSO 算法.计算机工程,2011,37(21):20–22.

7 孔艳,熊伟丽,高淑梅.一种单个粒子自适应修正的粒子群算法.计算机系统应用,2012,21(5):86–90.

8 Spall JC. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans. on Automatic Control, 1992, 37(3): 332–341.

9 Maeda Y, Kuratani T. Simultaneous perturbation particle swarm optimization. IEEE Congress on Evolutionary Computation, CEC'06. 2006. 672–676.

10 Kiranyaz S, Ince T, Gabbouj M. Stochastic approximation driven particle swarm optimization with simultaneous perturbation who will guide the guide? Applied Soft Computing, 2011, 11(2): 2334–2347.

11 Spall JC. Implementation of the simultaneous perturbation algorithm for stochastic optimization. Trans. on Aerospace and Electronic Systems, 1998, 34(3): 817–823.

12 朱海梅,吴永萍.一种高速收敛粒子群优化算法.控制与决策,2010,25(1):20–24.

13 Deep K, Bansal JC. Mean particle swarm optimisation for function optimization. International Journal of Computational Intelligence Studies, 2009, 1(1): 72–92.