一种带交叉因子的双向寻优粒子群优化算法*

温 雅, 李 国, 徐 晨
(深圳大学 数学与计算科学学院 智能计算科学研究所, 广东 深圳 518060)

摘 要: 针对传统粒子群算法易早熟、精度低、后期收敛速度慢等问题,结合反向学习理论,提出了一种基于交叉因子的双向寻优粒子群优化算法(CBMPSO)。该算法使初始种群在搜索区域均匀分布,计算粒子及其反向粒子的适应值,取最优作为初始种群;迭代过程增加对全局最差粒子的跟踪,随机开启基于交叉因子的双向学习机制。对几种典型函数的测试结果表明,CBMPSO 算法的寻优能力及收敛速度有了显著提高,并且能够有效避免早熟收敛问题。

关键词: 粒子群优化; 双向学习机制; 交叉因子; 早熟收敛
中图分类号: TP751.1; TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2013)01-0082-04
doi:10.3969/j.issn.1001-3695.2013.01.019

Particle swarm optimization of corro-factor and bilingual learning mechanism

WEN Ya, LI Guo, XU Chen
(*Institute of Intelligent Computing Science, College of Mathematics & Computational Science, Shenzhen University, Shenzhen Guangdong* 518060, *China*)

Abstract: To overcome the disadvantages of particle swarm optimization (PSO) algorithm such as premature, bad convergence rate, this paper presented an improved algorithm (CBMPSO). The algorithm first made the initial population in the searching space evenly distributed, then calculated the initial and the opposite ones' fitness value, chose the better ones as the initial population. Added global poorest position to the update of particle position and started corro-factor and bilingual learning Mechanism randomly. The numeric experiments indicate that the new strategy can not only speed up the convergence but also can avoid the premature convergence problem effectively.

Key words: particle swarm optimization(PSO); bilingual learning mechanism; corro-factor; premature convergence

0 引言

粒子群优化算法由于原理简单、可调参数少、易于实现等优点,迅速得到了广泛的应用。但是粒子群算法同时也存在易早熟、后期收敛速度慢等缺点。为了克服这一缺点,人们提出许多改进的算法。典型的改进算法可以分为以下两类:a)在基本粒子群算法中引入惯性权重参数 ω,并令其线性递减或自适应变化[1,2],有效平衡了全局搜索和局部搜索,这类改进方法能够提高收敛速度和收敛精度,但是无法克服早熟问题;b)采用多算法融合,如采用混沌搜索进行局部优化[3,4]等,这类算法综合了多种算法的优点,对改进算法的性能有较好的效果,但是仍然不能克服粒子群算法的早熟和多样性下降过快的问题。

既要保证算法的性能、提高收敛速度和收敛精度,又要尽可能地解决早熟问题,为此,笔者重点从初始种群与进化机制两个方面对算法进行改进,提出了带交叉因子的双向寻优粒子群优化算法。

1 PSO 算法

PSO 初始化为一群随机粒子(随机解),然后通过迭代找到最优解。在每一次迭代中,粒子通过跟踪两个极值来更新自己:a)粒子本身所找到的最优解,这个解叫做个体极值;b)整个种群目前找到的最优解,这个解叫做全局极值。

粒子在找到上述两个极值后,就根据式(1)和(2)来更新自己的速度和位置。

$$v_i^{d+1} = \omega v_i^d + c_1 \text{rand}(p_i^d - x_i^d) + c_2 \text{rand}(p_g^d - x_i^d) \quad (1)$$

$$x_i = x_i + v_i^d \quad (2)$$

其中:ω 是保持原来速度的系数,叫做惯性权重,一般取 0.9 ~ 0.1 之间的常数;c_1 是粒子跟踪自己历史最优值的权重系数,它表示粒子自身的认识,所以叫做"认知",通常设置为 2;c_2 是粒子跟踪群体最优值的权重系数,它表示粒子对整个群体知识

收稿日期: 2012-06-05; **修回日期**: 2012-07-11 **基金项目**: 国家自然科学基金资助项目(61070087);广东省教育部产学研结合项目(2009B090300355);深圳大学校级科研项目(4LGB)

作者简介: 温雅(1988-),女,河南泌阳人,硕士研究生,主要研究方向为智能计算(wenya122@126.com);李国(1967-),男,河南信阳人,副教授,主要研究方向为智能计算;徐晨(1965-),男,浙江海宁人,教授,博导,主要研究方向为小波计算在信息科学中的应用.

的认识,所以叫做“社会知识”,经常叫做“社会”,通常设置为2;rand 是[0,1]内均匀分布的随机数。

粒子的先前速度和自我认知部分体现了粒子对自身经验的积累;粒子的社会认知部分体现了粒子之间的信息传递。算法在随机初始化一群粒子之后迭代运行,并且在每次迭代过程中粒子通过式(1)(2)更新,直到满足迭代停止条件为止。

2　基于交叉因子与双向学习机制的粒子群优化算法

2.1　反向学习算法

反向学习算法是 Tizhoosh[5] 提出的一种新型的强化的学习算法。其基本思想是同时考虑变量的当前估计值与反向估计值,通过比较获得当前的最优值。接下来给出反向点与反向学习算法的定义。

定义 1　$x\in[a,b]$中的任意实数,定义 $\hat{x}=a+b-x$ 为 x 的反向点。

定义 2　$X=[x_1,x_2,\cdots,x_D]$是 D 维空间中的一点,其中 $x_1,x_2,\cdots,x_D\in R$ 且 $x_i\in[a_i,b_i]$, $\forall i\in\{1,2,\cdots,D\}$,定义 $\hat{X}=[\hat{x}_1,\hat{x}_2,\cdots,\hat{x}_D]$为 X 的反向点,其中 $\hat{X}=[\hat{x}_1,\hat{x}_2,\cdots,\hat{x}_D]$中元素取为 $\hat{x}_i=a_i+b_i-x_i$。

定义 3　假定 $X=[x_1,x_2,\cdots,x_D]$是 D 维空间中的一点,$f(\cdot)$是衡量该点的适应值函数,$\hat{X}=[\hat{x}_1,\hat{x}_2,\cdots,\hat{x}_D]$为 $X=[x_1,x_2,\cdots,x_D]$的反向点。如果 $f(\hat{X})<f(x)$,那么就用$\hat{X}$替代 X;否则继续采用点 X。

2.2　基于交叉因子的反向学习算法

在反向学习算法中引入交叉因子,这里的交叉因子指的是在计算粒子 X_i 的反向点时,随机地选择一个粒子 X_j,其中 $i\neq j$ 且 $i,j=1,2,\cdots,N$。将粒子 X_i 与 X_j 交叉合为一个粒子。该方法可以增加粒子的多样性,提高粒子向新区域学习的能力。基于交叉因子反向点的定义为

$$\hat{x}_i=\begin{cases}a_i+b_i-(m_1x_i+m_2x_j) & \text{rand}<p\\ x_i & \text{其他}\end{cases}\tag{3}$$

其中:rand、m_1、m_2 为[0,1]内的随机数,且 $m_1+m_2=1$;p 为算法中转入反向学习算法的概率。文献[6]中讨论了 p 的取值,通过测试显示,p 在取 0.1、0.3、0.5 时算法能取得较好结果。

2.3　基于交叉因子与双向学习机制的粒子群优化算法

Shi 等人[7] 在对粒子群优化(PSO)算法进行研究时发现,该算法在初期搜索效率很高,但在接近最优解时搜索速度降低,并且很难得到精确最优解。这是因为算法在后期陷入了局部最优。产生局部最优的原因可以从如下两个方面来解释,并试图从解释中找到增强 PSO 寻优性能的方法:

a)从数值仿真实验来看,粒子陷入局部最优表现为全局最优位置连续很多代变化很小或者没有变化,这说明种群向全局最优区域的搜索能力丧失。笔者认为初始种群相对集中是导致这一现象的原因之一,一旦初始种群相对集中,每次迭代的搜索区域就会相对较小,粒子将很容易陷入局部最优。

b)从社会心理学角度来看,粒子陷入局部最优是因为对于种群中的某个粒子来说,随着粒子自身的不断进化与成长,对信息的需求也逐渐增大,单依赖于 p_i、p_g 这两个极值已经不能满足粒子对于信息的需求,所以此时应当增强种群中其他粒子探索新区域的能力。根据以上分析,本文提出了基于交叉因子和双向学习机制的粒子群优化算法。

2.3.1　初始解的生成

初始解是算法进行搜索的起点,组成初始群体的个体在备选解空间中的分布状况会对算法搜索性能产生重大影响。在之前的粒子群算法中,初始群体生成是随机的,即随机产生若干个个体组成初始种群。文献[8]给出了一种基于小区间的初始解生成法,即先把各待优化参数的取值范围分成群体总数个小区间,再在各小区间中分别随机生成一个初始个体,使初始种群在整个解空间中均匀分布。

本文在此基础上进行改进,在实际进化算法中,如果能选择一些靠近最优个体的初始粒子,则在一定程度上可以加快算法的收敛速度。因此在计算上述分布于小区间的初始粒子适应度的同时,计算该粒子对应的基于交叉因子的反向个体的适应值,通过两者比较,选择更靠近最优个体的初始个体作为初始种群,这样既保证了初始群体含有较丰富的模式,又有利于整体的进化收敛速度。

小区间通常取等值区间,即将各优化参数的取值范围均匀划分为 n 个小区间,n 为种群规模,假设其中的一个小区间为$[a_i,b_i]$,$x_i\in[a_i,b_i]$,且 x_i 为随机数,则 $\hat{x}_i=a_i+b_i-(m_1x_i+m_2x_j)$,其中 $i\neq j$ 且 $j=1,2,\cdots,n$;m_1、m_2 为[0,1]内的随机数,且 $m_1+m_2=1$。

2.3.2　基于交叉因子与双向学习机制的粒子群优化算法

随机粒子的每次迭代过程中,在跟踪个体最优 p_i、全局最优 p_g 的前提下还要跟踪两个新的极值,一个是粒子本身的最差解 pp_i;另一个是整个种群目前出现的最差解 pp_g,并在迭代机制中引入全局最差 pp_g 的反向粒子$\widehat{pp_g}$作为新的学习因子,具体迭代机制如式(4)所示。

$$v_i^{d+1}=\omega v_i^d+c_1\text{rand}(p_i^d-x_i^d)+c_2\text{rand}(p_g^d-x_i^d)+c_3\text{rand}(\widehat{pp_g}-x_i^d)\tag{4}$$

其中:$\omega=\omega_{\max}-\dfrac{\text{iter}}{\text{iter}_{\max}}(\omega_{\max}-\omega_{\min})$,$c_1=c_2=1.5$,$c_3=1$。实验证明,$\widehat{pp_g}$不一定总是引导粒子群向最优值靠拢,即存在一些情况会影响到粒子的全局学习能力,故在此对基于交叉因子的反向点$\widehat{pp_g}$作了进一步改进:

$$\widehat{pp_g}=\begin{cases}a+b-(m_1pp_g+m_2p_i) & \text{rand}<p\\ 0 & \text{其他}\end{cases}\tag{5}$$

即新的迭代机制中$\widehat{pp_g}$以 p 的概率起作用。该方法一方面可以增加最优解的多样性,丰富学习内容;另一方面可以避免过度学习现象的出现。

2.3.3　CBMPSO 算法收敛性分析

文献[9]中给出了一种闭环控制的 PSO 动力学方法和 PSO 算法的收敛条件,在此本文对 CBMPSO 算法的收敛性进行验证。由 CBMPSO 算法推导出的线性系统状态方程如式

(6)所示。

$$\begin{bmatrix} v_i^{n+1} \\ x_i^{n+1} \end{bmatrix} = \tilde{A}\begin{bmatrix} v_i^n \\ x_i^n \end{bmatrix} + \tilde{B}\begin{bmatrix} p_i^n \\ p_g^n \\ \overset{\frown}{pp_g} \end{bmatrix} \tag{6}$$

其中:

$$\tilde{A} = \begin{bmatrix} w & -\varphi \\ w & 1-\varphi \end{bmatrix}, \tilde{B} = \begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 \\ \varphi_1 & \varphi_2 & \varphi_3 \end{bmatrix} \tag{7}$$

$$\varphi_1 = C_1 \text{rand}_1(), \varphi_2 = C_2 \text{rand}_2(), \varphi_3 = C_3 \text{rand}_3() \tag{8}$$

$$\varphi = \varphi_1 + \varphi_2 + \varphi_3 \tag{9}$$

由文献[9]中的结论可知,当 $0 < w < 1$ 时,若满足

$$w + 1 - 2\sqrt{w} < \varphi < w + 1 + 2\sqrt{w}$$

则系统稳定,算法收敛。本文 w 采用随进化代数的增加线性递减的方式,取值为[0.4,0.9],若取 $C_1 = C_2 = 1.5, C_3 = 1$,则 φ 在(0,4)区间变化,保证了算法的收敛性。

算法的基本步骤如下:

a)初始化。对搜索区域按维均匀划分,在所得的 n 个小区域上随机取值,得到 n 个粒子;计算每个粒子及其反向粒子的适应值,并进行比较,取适应值较小者组成初始种群。

b)迭代机制。每次迭代,寻找全局最优 p_g、全局最差粒子粒子 pp_g。

c)交叉因子。计算 pp_g,并用 rand 对反向粒子 pp_g_f 是否起作用进行选择。

d)按式(4)(5)对粒子的位置和速度进行更新。

e)若未达到终止条件,则转步骤 b)。

3 数值实验与结论

选取六个基本测试函数对 LDWPSO[7]、CPSO[10]、CBMPSO 的性能进行对比。

函数 1:$\sum_{i=1}^{n} x_i^2$, $-100 \leqslant x_i \leqslant 50$,该函数在 x_i 时取得最小值 0。

函数 2:$\sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor^2$, $-100 \leqslant x_i \leqslant 50$,该函数在 $x_i \in (-1.5, 0.5)$时取得最小值 0。

函数 3:$\sum_{i=1}^{n} (i+1) x_i^4$, $-100 \leqslant x_i \leqslant 50$,该函数在 x_i 时取得最小值 0。

函数 4:$\frac{\pi}{n}(10\sin^2 \pi y_1 + \sum_{i=1}^{n-1} (y_i - 1)^2 (1 + 10\sin^2 \pi y_{i+1}) + (y_n - 1)^2)$, $-100 \leqslant y_i \leqslant 50$,该函数在 $y_i = 1$ 时取得最小值 0。

函数 5:$\frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \frac{x_i}{\sqrt{i}} + 1$, $-100 \leqslant x_i \leqslant 50$,该函数在 $x_i = 0$ 时取得最小值 0。

函数 6:$\sum_{i}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$, $-100 \leqslant x_i \leqslant 50$,该函数在 $x_i = 1$ 时取得最小值 0。

在实验中,每个函数的参数设置如表 1 所示。

在实验中算法的公共参数设置为:种群大小固定为 300,粒子的最大速度为搜索区间长度的 20%,LDWPSO 算法中权重的变化范围为[0.9,0.4],CBMPSO 中的 p 取 0.1。本实验中每个函数的迭代次数为 1 000,运行次数为 50,实验结果如表 2、3 所示。若函数在指定迭代次数内没有达到预设精度,则指定迭代次数为收敛次数;否则第一次满足预设精度的迭代次数为收敛次数,函数运行时间为从程序开始执行到达到收敛次数的时间。表 2、3 中的收敛次数及运行时间均为 50 次实验的平均值,收敛率为满足预设精度的实验次数与实验总次数的比值。

表 1 实验参数设置

函数	种群大小	搜索区间	预设精度
1	300	[-100,50]	<0.01
2	300	[-100,50]	<0.01
3	300	[-100,50]	<0.01
4	300	[-100,50]	<0.01
5	300	[-100,50]	<0.01
6	300	[-100,50]	<0.02

表 2 LDWPSO、CPSO、CBMPSO 算法在最优值、收敛速度和成功率上的比较(30 维)

函数	算法	最优值	平均值	方差	平均收敛次数	收敛率/%	运行时间/s
1	LDWPSO	0.142348955	0.546912795	0.123165	1000	0	27.9
	CPSO	18.94238868	47.47153799	2458.711	1000	0	10.3
	CBMPSO	2.61E-25	1.26694E-25	1.52E-50	340	100	5.4
2	LDWPSO	0.261118453	0.79874042	0.774683	1000	0	29.4
	CPSO	17947.71962	29420.75869	4.95E+08	1000	0	13.5
	CBMPSO	2.37E-43	8.76488E-41	1.69E-80	281	100	4.9
3	LDWPSO	0.021548937	0.068837582	0.002817	1000	0	30.2
	CPSO	136.8703556	67.04184095	4138.054	1000	0	14.2
	CBMPSO	1.43E-25	1.22181E-25	9.94E-52	360	100	5.7
4	LDWPSO	308.2168066	634.2641335	79948.26	1000	0	28.9
	CPSO	807.0179922	969.2694325	7042.372	1000	0	6.9
	CBMPSO	3.34E-24	1.43701E-23	2.71E-46	262	100	4.1
5	LDWPSO	0.001942866	0.03059899	0.001211	875	75	30.1
	CPSO	0.193514371	0.396404722	0.109173	753	79	11.8
	CBMPSO	0	1.11022E-16	1.23E-32	58	100	3.1
6	LDWPSO	299.9106225	1284.649923	728394.9	1000	0	25.8
	CPSO	24943.58119	191713.4255	6.6E+10	1000	0	4.7
	CBMPSO	8.10E-13	2.09859E-08	1.32E-15	639	100	5.3

表 3 LDWPSO、CPSO、CBMPSO 算法在最优值、收敛速度和成功率上的比较(50 维)

函数	算法	最优值	平均值	方差	平均收敛次数	收敛率/%	运行时间/s
1	LDWPSO	0.544541927	10.74723072	139.9878	1000	0	40.1
	CPSO	7.673508098	28.00854211	378.4776	1000	0	15.7
	CBMPSO	2.20E-12	9.41977E-12	1.08E-22	458	100	10.5
2	LDWPSO	41.87287677	431.8393662	369250.2	1000	0	42.5
	CPSO	12259.91778	17834.33677	2.7E+08	1000	0	20.9
	CBMPSO	2.98E-19	1.42164E-17	5.29E-34	344	100	8.2
3	LDWPSO	0.143132623	13.64073918	538.208	1000	0	46.3
	CPSO	11.20337256	58.8024027	2848.099	1000	0	23.9
	CBMPSO	1.31E-12	2.23508E-11	6.47E-22	448	100	7.6
4	LDWPSO	383.1939351	650.2520848	48173.33	1000	0	44.7
	CPSO	374.0707019	822.3214201	142015	1000	0	10.2
	CBMPSO	6.56E-12	5.67055E-11	4.42E-21	321	100	6.3
5	LDWPSO	0.100241053	0.174637964	0.007198	963	69	44.1
	CPSO	0.273785414	0.586963208	0.163121	876	72	15.7
	CBMPSO	3.16E-12	4.16076E-12	6.45E-24	126	100	5.1
6	LDWPSO	381.5385082	19884.54619	2.59E+08	1000	0	39.3
	CPSO	246291.6668	187307.6593	4.58E+10	1000	0	6.1
	CBMPSO	0.000103678	0.001998268	1.91E-06	942	100	7.5

方案 1 固定粒子的维数为 30,迭代次数为 1 000 次,每个函数运行 50 次,以 50 次运行结果的平均值和最小值作为算法

的优化结果和最优值,并计算相应的方差。实验结果见表 2,函数 1 ~6 在维数 =30、迭代次数 =1 000 的进化曲线如图 1 ~ 6 所示。

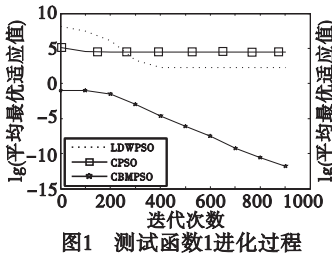

图1　测试函数1进化过程

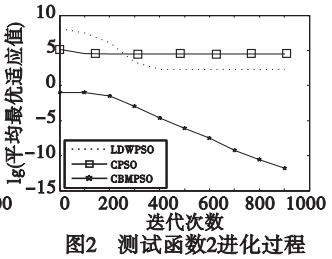

图2　测试函数2进化过程

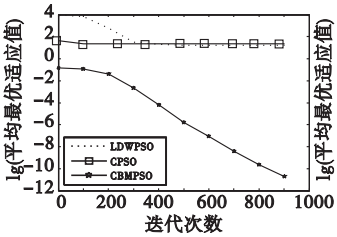

图3　测试函数3进化过程

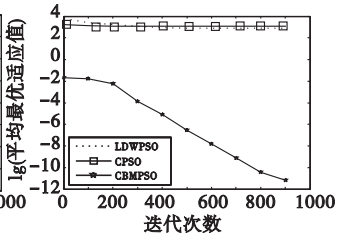

图4　测试函数4进化过程

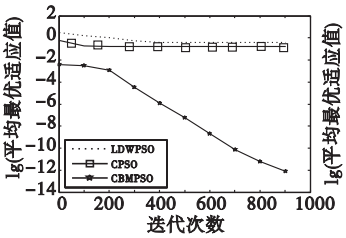

图5　测试函数5进化过程

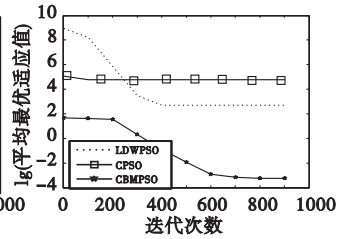

图6　测试函数6进化过程

方案2　固定粒子的维数为50,迭代次数为1 000 次,每个函数运行 50 次,以 50 次运行结果的平均值和最小值作为算法的优化结果和最优值,并计算相应的方差。实验结果如表 3 所示。

由上述进化曲线可以看出,由于新算法对初始种群的改进,CBMPSO 在初始时刻的进化就领先于之前的两种算法,而在后期 CBMPSO 的收敛速度也比 LDWPSO 和 CPSO 要快很多,后期不易陷入局部最优,故可以得出如下结论:

a)收敛速度。CBMPSO 算法对于初始种群的改进使得粒子群的收敛速度得到了很大的提升,使初始种群均匀分布是为了增加初始种群的多样性,反向粒子的引入则使得初始种群接近最优值的可能性进一步加大,从而迅速向最优值收敛。

b)收敛精度。在上述结果中,CBMPSO 算法以相对较少的迭代次数达到了预设精度,交叉因子的引进进一步增加了粒子的多样性,同时增强了粒子向新区域学习的能力,有效地平衡了局部与全局的搜索能力,使得粒子一旦开始迭代就可以迅速达到最优。

c)解决早熟问题。每次迭代过程中,在追踪全局最优的同时也追踪全局最差粒子的反向粒子,使粒子以一定的概率开启反向学习机制,有效地避免了全局极值的过度单一,及时解决了早熟问题。

4　结束语

粒子群算法存在后期收敛速度慢、容易陷入局部最优等缺点,笔者认为好的初始化方法可以加快整个寻优过程,全面的学习机制可以增强粒子探索新区域的能力,有效地避免粒子陷入局部最优。鉴于此,本文重点从种群的初始化方法和迭代机制两个方面对粒子群优化算法进行改进:a)引入反向粒子的概念,通过找到更接近最优个体的初始粒子来加快算法的收敛速度;b)每次迭代过程中追踪全局最优的同时追踪全局最差粒子的反向粒子,为了增加最优解的多样性,同时为了避免粒子群过度向该反向粒子学习,引入交叉因子,使该反向粒子以一定的概率在全局寻优过程中起作用。实验证实,基于交叉因子与双向学习机制的粒子群优化算法收敛速度快、跳出局部最优能力强、寻优精度高。

参考文献:

[1] 黄泽霞,俞攸红,黄德才. 惯性权自适应调整的量子粒子群优化算法[J]. 上海交通大学学报,2012,46(2):228-232.

[2] 徐刚,杨玉群,刘斌斌. 一种非线性自适应粒子群优化算法[J]. 南昌大学学报:理科版,2011,35(4):323-326.

[3] 赵志刚,常成. 自适应混沌粒子群优化算法[J]. 计算机工程,2011,37(15):128-130.

[4] 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):24-30.

[5] TIZHOOSH H R. Opposition-based learning:a new scheme for machine intelligence[C]//Proc of International Conference on Computional Intelligence for Modeling, Control and Automation. 2005:695-701.

[6] WANG Hui,LI Hui,LIU Yong,*et al*. Opposition-based particle swarm algorithm with cauchy mutation [C]//Proc of IEEE Congress on Evolutionary Computation. 2007:4750-4756.

[7] SHI Y,EBERHART R. A modified particle swarm optimizer[C]//proc of IEEE Intenational Conference on Evolutionary Computation, Proceedings. 1998:69-73.

[8] 丁海军,冯庆娴. 基于 boltzmann 选择策略的人工蜂群算法[J]. 计算机工程与应用,2009,45(31):53-56.

[9] SAMAL N R,KONAR A,DAS S,*et al*. A closed loop stability analysis and parameter selection of the particle awarm optimization dyna-mics for faster convergebce[C]//Proc of IEEE Congress on Evolutionary Computation. 2007:1769-1776.

[10] CLERC M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization[C]//Proc of Congress on Evolutionary Computation. Piscataway,NJ:IEEE Press,1999:1951-1957.

[11] CHEN Dong,WANG Gao-feng,CHEN Zhe-yi. The inertia weight self-adapting in PSO[C]//Proc of the 7th World Congress on Intelligent Control and Automation. 2008:5313-5316.

[12] FENG C S,CONG S,FENG X Y. A new adaptive inertia weight strategy in particle swarm optimization[C]//Proc of IEEE Congress on Evolutionary Computation. 2007:4186-4190.

[13] LIU Bo,WANG Ling,JIN Yi-hui,*et al*. Improved particle swarm optimization combined with chaos [J]. Chaos Solitons & Fractals, 2005,25(5):1261-1271.

[14] 黄发良,郑小建. 含多 leader 交叉算子的粒子群优化算法[J]. 小型微型计算机系统,2012,31(1):130-134.

[15] 王燕,贺兴时,王凡,等. 改进反向粒子群算法及其在噪声中的应用[J]. 西安工程大学学报,2011,25(5):721-725.