文章编号: 1001-0920(2011)04-0548-05

基于生物寄生行为的双种群粒子群算法

秦全德, 李荣钧

(华南理工大学 工商管理学院，广州 510640)

摘 要: 在分析生物共生关系的基础上, 将兼性寄生行为的机制嵌入粒子群算法中, 构建了一种由宿主群和寄生群两个种群组成的粒子群算法——PSOPB. 该算法中两个种群间隔一定的迭代次数并按个体适应度的大小排序, 相互交换粒子. 为了体现“优胜劣汰”的生物进化法则, 宿主群中适应度较差的一半粒子被淘汰, 而由重新初始化的粒子代替以维持群体规模不变. 标准测试函数的仿真结果表明了 PSOPB 算法的有效性.

关键词: 粒子群算法；寄生行为；双种群

中图分类号: TP18 **文献标识码:** A

Two-population particle swarm optimization algorithm based on bio-parasitic behavior

QIN Quan-de, *LI Rong-jun*

(School of Business Administration，South China University of Technology，Guangzhou 510640，China. Correspondent：QIN Quan-de，E-mail：qinquande@gmail.com)

Abstract: Based on the analysis of biological symbiotic relationship, the mechanism of facultative parasitic behavior is incorporated into the particle swarm optimization(PSO) to propose a two-population PSO model called particle swarm optimization based on parasitic behavior(PSOPB), which consists of the host and the parasite population. In this model, the two populations exchange particles according to individual's fitness value sorted in a certain number of iterations. The parasitic population gets the particles with good fitness, whereas the particles with poor fitness belong to the host. In order to embody the rule of survival of the fittest in biological evolution, the particles with poor fitness in the host population are removed and replaced by the re-initialization of the particles in order to maintain constant population size. The experiment results of some benchmarks show the effectiveness of PSOPB.

Key words: particle swarm optimization algorithm；parasitic behaviour；two populations

1 引 言

粒子群优化(PSO)算法是模拟鸟群觅食行为的规律而提出来的一种智能优化方法[1]. PSO 算法需要调整的参数少, 收敛速度快, 且易于编程实现[2]. 近些年 PSO 算法快速发展, 已成功地应用于函数优化、神经网络训练和金融优化等众多领域[3-5].

Angeline[6] 于 1998 年指出, PSO 算法在搜索初期收敛速度较快, 但在搜索后期易早熟, 陷入局部最优点. 针对这一问题, 人们给出了多种改进方式, 包括算法的参数调节[7], 拓扑结构的改进[8], 与其他算法的混合[9]等.

PSO 算法源于对社会型群居动物的行为模拟, 因而将大自然中的一些生物行为融入 PSO 算法是一条潜在可行的改进途径. 国内外学者在这方面开展了一些研究. 文献 [10] 在基于生物生命周期现象的基础上, 提出一种新的 PSO 算法, 该算法将每一次迭代执行过程视为生物个体一个完整的生命周期. 受生物中捕食猎物的协同进化启发, Silva[11] 提出了 Predator-Prey 的 PSO 算法模型. 在该模型中粒子分成两类: Predator 与 Prey. Predator 粒子在搜索过程中迫使陷入局部最优点的粒子逃离, 而 Prey 粒子则受 Predator 粒子的排斥逐步靠近全局最优解. He 等人[12] 根据动物群中的被动聚众现象提出了一种新型粒子群优化算法 PSOPC (particle swarm optimizer with passive congregation). PSOPC 算法的速度更新不仅考

收稿日期: 2010-01-21；**修回日期:** 2010-04-12.

基金项目: 国家自然科学基金项目(71071057)；高等学校博士学科点专项基金项目(20060561002).

作者简介: 秦全德(1979–), 男, 博士生, 从事智能计算及应用的研究；李荣钧(1946–), 男, 教授, 博士生导师, 从事智能计算、系统优化等研究.

虑了个体的经验和邻居中最好个体的经验, 同时还考虑了群体中其他同伴的经验. Niu[13] 依照细菌的趋化行为, 提出了一种改进粒子群算法. 在该算法中粒子不仅被当前的最优位置和自身的最优位置吸引, 同时也会被自己历史最差位置和群体最差位置所排斥. 刘金洋等人[14] 将大雁在迁徙过程中的飞行机制引入标准 PSO 中, 提高了算法的性能.

目前从已有文献来看, 从生物行为对粒子群进行改进的研究还不够深入. 本文根据生物寄生行为的规律, 提出一种双种群的 PSO 改进算法——PSOPB (particle swarm optimization based on parasitic behavior). 在 PSOPB 中粒子分成寄生群和宿主群两个种群, 种群间模拟生物的寄生行为机制进行粒子的交换. 通过标准测试函数的仿真结果表明, 该算法具有较快的收敛速度和全局搜索能力.

2　生物寄生行为

德国真菌学家德贝里于 1879 年首先提出了共生的概念[15]. 生物之间的"共生"是指生物体之间生活在一起的交互作用. 共生的生物在生理上相互分工, 互换生命活动的产物, 在组织上形成新的结构. 从生物体利弊关系角度, 共生关系可分成互利共生、共栖和寄生[16]. 互利共生是指共生的生物体成员彼此都得到好处; 共栖是指仅对其中一方生物体有益, 对另一方则没有影响; 寄生是指一种生物寄生在另一种生物的体内或体表, 从那里吸取营养物质来维持生活[16]. 营寄生生活的生物称为寄生物, 被侵害的生物称为宿主. 一般来说, 寄生行为对寄生体是有利的, 而对宿主则是有害的. 寄生现象广泛存在于植物间、动物间和动植物间. 根据寄生程度的不同可分为两种: 离开寄生便完全不能生活的专性寄生; 以及既可以营寄生生活, 也可以营独立生活的兼性寄生.

大多数学者认为, 寄生物是从早期生物史中的自由生活生物进化而来的. 在生活方式转变的过程中, 寄生物和宿主相互适应、相互斗争, 从而达到寄生关系的成立[17]. 在二者协同进化的过程中, 宿主对寄生物具有先天性免疫和获得性免疫两种不同程度的免疫机制[18]. 寄生物侵入宿主后, 抗原物质刺激宿主免疫系统, 从而对寄生物可发挥杀伤效应, 而且对同种寄生虫的再感染具有一定抵抗力, 这是宿主获得性免疫的一般表现. 在大自然界, 脊椎动物被寄生物感染后会产生强烈的免疫反应, 如果再次遭遇同样病原体, 则免疫记忆会快速生产特异抗体, 提高免疫力. 植物和低等动物在受到感染后也能提高免疫力. 例如, 烟草植物的一片叶子被烟草花叶病毒感染后, 会提高整个植物体的防御性化学物质水平, 从而增加对多种病原体的抵抗力. 寄生物在与宿主长期相互适应过程中, 有些寄生物能逃避宿主的免疫效应, 这种现象称为免疫逃避[18].

寄生物和宿主之间长期的协同进化关系, 常会使寄生行为所带来的有害"负作用"减弱, 甚至演变为互利共生关系[19].

3　基于寄生行为的粒子群算法

3.1　粒子群算法基本框架

PSO 算法的基本思想是将每个粒子视为问题的一个可行解. 粒子 i 在 t 次迭代时的状态属性由两个向量描述: 位置向量 $x_i^t = [x_{i1}^t, x_{i2}^t, \cdots, x_{id}^t]$ 和速度向量 $v_i^t = [v_{i1}, v_{i2}, \cdots, v_{id}]$. 其中: $x_{id}^t \in (L_d, U_d)$, L_d 和 U_d 分别为搜索空间的下限和上限, d 为搜索空间的维数; $v_{id}^t \in [V_{\min}, V_{\max}]$, $V_{\min}$ 和 $V_{\max}$ 分别为粒子飞行的最小和最大速度. 粒子初始的位置和速度在相应的范围内初始化. 在算法的迭代过程中, 每个粒子的速度和位置将按下式更新:

$$v_{id}^{t+1} = \omega v_{id}^t + c_1 r_1(\mathrm{pB}_i^t - x_{id}^t) + c_2 r_2(\mathrm{nB}_i^t - x_{id}^t), \tag{1}$$

$$x_{id}^{i+1} = x_{id}^t + v_{id}^{t+1}, \tag{2}$$

其中 ω 为惯性权重. 在标准粒子群算法中, ω 按下式递减变化[20]:

$$\omega = (\omega_{\mathrm{start}} - \omega_{\mathrm{end}})\left(\frac{\mathrm{iter}_{\max} - \mathrm{iter}}{\mathrm{iter}_{\max}}\right) + \omega_{\mathrm{end}}, \tag{3}$$

一般取 $\omega_{\mathrm{start}} = 0.9$, $\omega_{\mathrm{end}} = 0.4$. 本文将这种惯性权重按式 (3) 线性递减的 PSO 算法称为 PSO-LIW (particle swarm optimization with linear-decreased inertia weight). 式 (1) 中, r_1 和 r_2 为均匀分布在 [0,1] 区间的随机数.

2002 年, Clerc[21] 采用压缩因子 λ 来研究 PSO 算法的收敛行为, 此时算法的速度更新依照下式进行:

$$v_{id}^{t+1} = \lambda(v_{id}^t + c_1 r_1(\mathrm{pB}_i^t - x_{id}^t) + c_2 r_2(\mathrm{nB}_i^t - x_{id}^t)). \tag{4}$$

本文将具有压缩因子的 PSO 算法称为 CPSO (constriction factor particle swarm optimization). 压缩因子 λ 按下式计算:

$$\lambda = \frac{2}{\left|2 - \varphi + \sqrt{\varphi^2 - 4\varphi}\right|}. \tag{5}$$

当 $\varphi = c_1 + c_2 = 4.1$ 时, $\lambda = 0.729$, 算法能较好地平衡开发和利用[21]. c_1 和 c_2 称为学习因子, 在 PSO-LIW 中, 一般设置 $c_1 = c_2 = 2$; 而在 CPSO 中, 通常设置 $c_1 = c_2 = 2.05$.

式 (1) 和 (4) 中的 pB_i^t 表示粒子 i 最优的个体位置, 记为 $\mathrm{pB}_i^t = [\mathrm{pB}_{i1}^t, \mathrm{pB}_{i2}^t, \cdots, \mathrm{pB}_{id}^t]$; nB_i^t 表示粒子 i 最好的邻居位置, 记为 $\mathrm{nB}_i^t = [\mathrm{nB}_{i1}^t, \mathrm{nB}_{i2}^t, \cdots, \mathrm{nB}_{id}^t]$. 本文采用全局版本的 PSO, nB 被群体最优位置 gB 所

代替.

3.2 PSOPB 算法模型

在本文提出的PSOPB 算法中, 将粒子分成两个种群: 一个称为寄生群, 其粒子数量为 N^P; 另一个称为宿主群, 粒子数量为 N^H. 两个种群之间的兼性寄生关系比较适合嵌入粒子群算法中, 即间隔一定的迭代次数 k, 寄生群从宿主群吸收一次营养. 发生寄生行为时, 将两个种群的粒子按照适应度进行排序, 并将适应度大于或等于排序为 $0.5(N^P+N^H)$ 的粒子归类到寄生群, 剩下的粒子划分到宿主群. 宿主被寄生物入侵后产生了获得性免疫, 在算法的模拟中表现为: 当宿主群的最优粒子比寄生群的最优粒子的适应度差时, 宿主群的粒子同时向个体最优、群体最优和寄生群的最优粒子3个方向飞行; 反之, 两个种群独立进化. 宿主群被寄生群吸收营养后, 将宿主群中适应度较差的粒子按一定比率进行淘汰, 从而体现"优胜劣汰"的进化法则. 为维持宿主群规模不变, 用重新初始化的粒子代替, 但粒子的个体最优位置维持不变.

综上分析, 在PSOPB 中寄生群的速度按照式 (4) 更新, 宿主群的速度则按下式更新:

$$v_{id}^{t+1}=\begin{cases}\lambda(v_{id}^t+c_{11}r_1(\mathrm{pB}_i^t-x_{id}^t)+\\ c_{12}r_2(\mathrm{gB}^H-x_{id}^t)+\\ c_{13}r_3(\mathrm{gB}^P-x_{id}^t)),\ \mathrm{fgB}^H<\mathrm{fgB}^P;\\ \lambda(v_{id}^t+c_1r_1(\mathrm{pB}_i^t-x_{id}^t)+\\ c_2r_2(\mathrm{gB}^H-x_{id}^t)),\ \mathrm{fgB}^H\geqslant\mathrm{fgB}^P.\end{cases}\tag{6}$$

其中: c_{11}, c_{12}, c_{13}, c_1, c_2 为学习因子; fgB^H 和 fgB^P 分别为宿主群和寄生群体中最优个体的适应度; r_3 为均匀分布在 [0,1] 之间的随机数; 其他参数的意义同3.1 节所述.

4 实验分析

4.1 测试函数

本文选取智能优化中 8 个常用的测试函数进行分析, 详细数据见表 1. 其中: $f_1\sim f_3$ 是单峰函数, $f_4\sim f_8$ 是多峰函数.

表 1 标准测试函数及其搜索和初始化范围

函数名称	函数数学表达式	搜索范围	初始化范围
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	$(-100,100)^n$	$(50,100)^n$
Schwefel's Problem 1.2	$f_2(x)=\sum_{i=1}^{n}\Big(\sum_{j=1}^{i}x_j\Big)^2$	$(-100,100)^n$	$(50,100)^n$
Rosenbrock	$f_3(x)=\sum_{i=1}^{n}\big(100(x_{i+1}-x_i)^2+(x_i-1)^2\big)$	$(-30,30)^n$	$(10,30)^n$
Schwefel's Problem 2.26	$f_4(x)=418.982\,9n-\sum_{i=1}^{n}\Big(x_i\sin(\sqrt{\lvert x_i\rvert})\Big)$	$(-500,500)^n$	$(100,500)^n$
Ackley	$f_5(x)=-20\exp\Big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\Big)-\exp\Big(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\Big)+20+e$	$(-32,32)^n$	$(10,20)^n$
Rastrigin	$f_6(x)=\sum_{i=1}^{n}\big(x_i^2-10\cos(2\pi x_i)+10\big)$	$(-10,10)^n$	$(2.56,5.12)^n$
Griewank	$f_7(x)=\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\Big(\frac{x-i}{\sqrt{i}}\Big)+1$	$(-600,600)^n$	$(300,600)^n$
Weierstrass	$f_8(x)=\sum_{i=1}^{n}\Big(\sum_{k=0}^{20}[a^k\cos(2\pi b^k(x_i+0.5))]\Big)-n\sum_{k=0}^{20}(a^k\cos(2\pi b^k\times0.5))$		
	$a=0.5,b=3$	$(-0.5,0.5)^n$	$(-0.5,0.2)^n$

4.2 实验环境设置

本文采用非对称的方法初始化, 即初始化范围的 $[X_{\min},X_{\max}]$ 取值为搜索空间的一部分. 对PSOPB, CPSO, PSOPC 和 PSO-LIW 设置的速度上下限分别为 $V_{\max}=U_d$, $V_{\min}=L_d$. PSOPB 的速度更新式 (6) 中, $c_{11}=c_{12}=c_{13}=1.367$, $c_1=c_2=2.05$. 寄生行为发生后宿主群淘汰一半粒子, 即 $\gamma=0.5$. PSOPC 的参数依据文献 [12] 选取, $\omega_{\mathrm{start}}=0.9$, $\omega_{\mathrm{end}}=0.7$, $c_1=c_2=0.5$, $c_3=0.4-0.6$. CPSO 和 PSO-LIW 的其余参数设置同 3.1 节所述. 测试函数的维数 n 均设为 30. 进行比较的各算法粒子的数量均设为 80, 其中在PSOPB中将寄生群和宿主群的粒子数目设置相同, 即 $N^H=N^p=40$.

在实际问题中, 一般需要对变量的取值范围进行限制. 本文采用文献 [22] 的方法对搜索边界进行处理, 粒子 i 在更新位置时, 如果 $X_{id}^t>U_d$ 或 $X_{id}^t<L_d$, 则 $V_{id}^t=0$, $X_{id}^t=\min(\max(X_{id}^t,L_d),U_d)$. 为了测试PSOPB 的性能对参数 k 值是否灵敏, 本文采用不同的 k 值对 f_3 和 f_6 的 30 维函数进行仿真实验. 实验设置的最大迭代次数为 3 000 次, 每个实验独立运行 15 次, 所得结果列于表 2. 从表 2 可以看出, 当 $k=20$ 时, 单峰函数 f_3 的性能较好; 当 $k=100$ 时, 多峰函数 f_6 的优化性能较好. 在本文仿真实验中, 优化单峰函数和多峰函数的 k 值分别设为 20 和 100.

表 2　不同 k 值下 f_3 和 f_6 的测试结果

函数名称	不同 k 值下求解结果的均值和标准差(括号内数值)			
	20	50	100	150
f_3	3.893 3 (4.180 3)	5.082 5 (5.562 1)	5.303 7 (4.213 1)	7.509 6 (4.938 2)
f_6	39.885 9 (8.700 9)	37.543 4 (9.902 3)	20.236 8 (7.044 5)	20.963 5 (7.468 5)

4.3　实验结果与分析

将进行比较的4种粒子群算法对8个测试函数分别独立运行25次, 最大迭代次数为6 000次, 得到的实验结果如表3所示. 图1～图4描述了每种算法对 f_2, f_3, f_4 和 f_6 优化过程中平均最优值(取10为底的对数)的变化曲线. 从表3和图1～图4可以看出, 与其他算法相比, PSOPB对于单峰和多峰函数都体现了较快的收敛速度和更高的搜索精度. 在单峰函数中, PSOPB的优化性能明显优于PSO-LIW和PSOPC. 在 f_1 和 f_2 的优化中, PSOPB的性能优于CPSO. 在极难优化的"香蕉函数" f_3 中, PSOPB的搜索精度远好于CPSO, 且持续寻优能力强, 不容易停滞. f_4 为不可分离的多峰函数, 具有一定的欺骗性, 由于全局最优与最好的局部最优相距甚远, 导致搜索算法往往朝着错误的方向进行收敛. 在对 f_4 的优化过程中, PSOPB比PSO-LIW和CPSO具有更快的收敛速度和搜索精度. 在迭代前期, PSOPB的收敛速度较慢于PSOPC; 但在迭代后期, 其搜索精度高于PSOPC. 从对 f_5, f_6, f_7 和 f_8 的仿真结果可以看出, PSOPB体现出很好的优化效果, 特别是对于具有大量局部最优点的复杂多峰函数 f_6, PSOPB不容易早熟.

从以上分析可以发现, 对于一般粒子群算法难以优化的 f_3, f_4 和 f_6, PSOPB表现出了优秀的性能.

表 3　各种算法的测试性能比较结果

函数名称	函数维数	最大迭代次数	指标	比较的PSO算法			
				PSO-LIW	CPSO	PSOPB	PSOPC
f_1	30	6 000	最优值	1.810 6e-050	4.662 1e-150	2.032 1e-152	1.786 4e-124
			最差值	2.817 3e-044	1.027 8e-146	1.128 3e-146	4.994 8e-100
			平均值	2.079 7e-045	2.564 7e-145	1.154 3e-147	9.989 6e-100
			标准差	7.232 3e-045	3.877 7e-145	3.559 8e-147	2.233 7e-100
f_2	30	6 000	最优值	0.163 8	1.056 9e-019	6.766 7e-023	0.166 6
			最差值	1.511 6	6.086 1e-017	2.892 9e-019	1.392 7
			平均值	0.589 7	1.099 1e-017	6.451 2e-020	0.668 63
			标准差	0.444 8	1.836 7e-017	1.071 1e-019	0.533 6
f_3	30	6 000	最优值	1.951 7	0.002 5	1.935 8e-006	16.695 2
			最差值	78.442 3	11.626 3	4.009 9	29.028 3
			平均值	30.725 2	6.197 1	0.478 5	23.566 4
			标准差	26.135 8	3.802 2	1.245 2	4.403 7
f_4	30	6 000	最优值	0.476 8e+03	0.651 4e+03	0.000 4	0.000 4
			最差值	2.025 6 e+03	1.482 0e+03	373.753 7	651.419 4
			平均值	1.056 2e+03	0.983 4e+03	141.497 3	268.464 7
			标准差	0.423 7 e+03	0.271 5e+03	177.145 5	273.715 6
f_5	30	6 000	最优值	4.537 4e-017	6.217 2e-015	2.664 5e-015	4.712 5e-015
			最差值	7.322 8e-014	1.778 2	6.217 2e-015	2.430 1e-013
			平均值	6.217 2e-015	0.476 5	5.862 4e-015	3.108 6e-014
			标准差	5.415 3e-015	0.774 4	1.123 5e-015	4.864 3e-014
f_6	30	6 000	最优值	7.959 7	33.829 5	4.974 8	21.889 6
			最差值	28.849 2	69.647 2	31.839 5	38.803 2
			平均值	19.833 3	43.679 6	11.737 2	27.662 1
			标准差	5.351 5	11.407 3	7.973 2	6.726 1
f_7	30	6 000	最优值	0	0	0	0
			最差值	0.063 8	0.029 6	0.017 2	0.021 3
			平均值	0.015 0	0.011 8	0.004 9	0.009 8
			标准差	0.022 1	0.010 7	0.005 9	0.007 1
f_8	30	6 000	最优值	0	0.272 9	0	0.298 8
			最差值	1.504 2	5.620 3	0.028 9	4.182 0
			平均值	0.134 6	3.191 6	0.004 8	1.828 0
			标准差	0.374 2	1.999 7	0.011 8	1.185 8

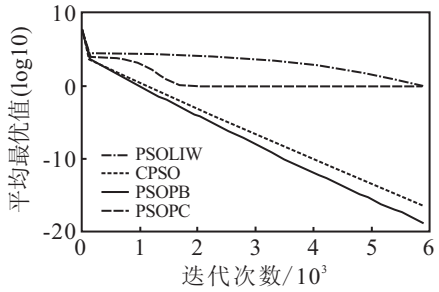

图 1 各种算法 f_2 的平均最优值变化曲线

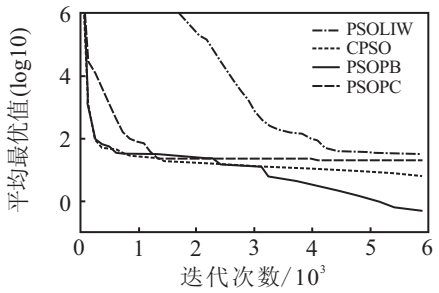

图 2 各种算法 f_3 的平均最优值变化曲线

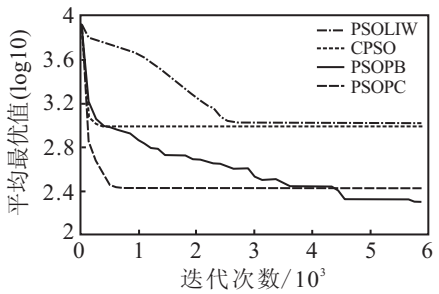

图 3 各种算法 f_4 的平均最优值变化曲线

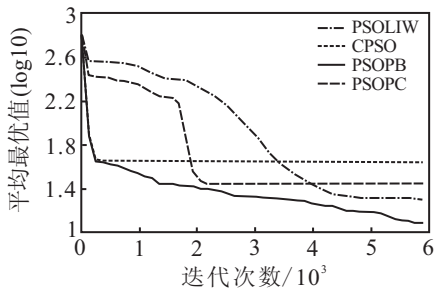

图 4 各种算法 f_6 的平均最优值变化曲线

5 结 论

本文构建了一种新的基于生物行为机制改进的粒子群算法——PSOPB. 该算法模型由寄生群和宿主群两个种群构成, 种群之间定期模拟生物的寄生行为并按照适应度的大小交换粒子. 实验分析表明, 单峰函数间隔20代交换粒子其性能较佳. 而对于多峰函数, 间隔100代可体现其良好的优化效果. 两个种群交换粒子后, 宿主群中适应度较低的一半粒子被淘汰, 体现了“优胜劣汰”的生物进化法则. 为维持群体规模不变, 用重新初始化的粒子代替, 可在一定程度上保证群体的多样性. 对标准测试函数的实验表明, 两个种群之间这种协同进化加快了算法的收敛速度和搜索精度. 本文未将寄生物的免疫逃避行为嵌入算法中, 这与实际的寄生行为不吻合. 进一步的研究重点是如何嵌入寄生物的免疫逃避机制, 以及将PSOPB应用于经济管理和工程实际的优化问题中, 扩展其应用范围.

参考文献(References)

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

[2] 谢晓锋, 张文俊, 杨之廉. 微粒群算法综述[J]. 控制与决策, 2003, 18(2): 129-133.
(Xie X F, Zhang W J, Yang Z L. Overview of particle swarm optimization[J]. Control and Decision, 2003, 18(2): 129-133.)

[3] Shi Y, Eberchart R C. A modified particle swarm optimizer[C]. IEEE Congress on Evolutionary Computation. Anchorage, 1998: 69-73.

[4] Mendes R, Cortez P, Rocha M, et al. Particle swarms for feed forward neural network training[C]. Int Joint Conf on Neural Networks. Seatle, 2002: 1895-1899.

[5] Cura T. Particle swarm optimization approach to portfolio optimization[J]. Nonlinear Analysis: Real World Applications, 2009, 10(4): 2396-2406.

[6] Angeline P J. Evolutionary optimization versus particle swarm optimization and philosophy and performance difference[C]. Proc of 7th Annual Conf on Evolutionary Programming. San Diego, 1998: 601-610.

[7] Shi Y, Eberhart R C. Fuzzy adaptive particle swarm optimization[C]. Proc of the IEEE Congress on Evolutionary Computation. Seoul, 2001: 79-85.

[8] Suganthan P. Particle swarm optimizer with neighborhood operator[C]. Proc of the IEEE Congress of Evolutionary Computation. Washington DC, 1999: 1958-1961.

[9] Angeline P J. Using selection to improve particle swarm optimization[C]. Proc of IEEE World Congress on Computational Intelligence. Anchorage, 1998: 84-89.

[10] Krink T, Lovbjerg M. The life cyclemodel: Combining particle swarm optimisation, genetic algorithms and hillclimbers[C]. Lecture Notes in Computer Science. Berlin: Springer, 2002, 2439: 621-630.

[11] Silva, Neves A, Costa E. An empirical comparison of particle swarm and predator prey optimisation[C]. Lecture Notes in Artificial Intelligence. Berlin: Springer, 2002, 2464: 103-110.

[12] He S, Wu Q H, Wen J Y, et al. A particle swarm optimizer with passive congregation[J]. BioSystems, 2004, 78: 135-147.

(下转第557页)