文章编号: 1001-0920(2009)09-1406-06

改进的多种群协同进化微粒群优化算法

陶新民[1], 徐 晶[2], 杨立标[1], 刘 玉[1]
(1. 哈尔滨工程大学 信息与通信工程学院, 哈尔滨 150001;
2. 黑龙江科技学院 数力系, 哈尔滨 150027)

摘 要: 提出一种改进的基于多种群协同进化的微粒群优化算法(PSO). 该算法首先利用免疫算法实现解空间的均匀划分, 增加了算法稳定性和全局搜索能力. 在运行过程中, 通过种群进化信息生成解优胜区域, 指导变异生成的微粒群向最优解子空间逼近, 提高算法逃出局部最优的能力. 将此算法与 PSO 算法和多种群协同进化微粒群算法进行比较, 数据实验证明, 该算法不仅能有效地克服其他算法易陷入局部极小值的缺点, 而且全局收敛能力和稳定性均有显著提高.

关键词: 粒子群算法; 多种群协同进化; 免疫算法; 优胜区域
中图分类号: TP18 **文献标识码**: A

Multi-species cooperative particle swarm optimization algorithm

TAO Xin-min[1], *XU Jing*[2], *YANG Li-biao*[1], *LIU Yu*[1]
(1. College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China;
2. Department of Mathematics and Mechanics, Heilongjiang Institute of Science and Technology, Harbin 150027, China. Correspondent: TAO Xin-min, E-mail: taoxinmin@hrbeu.edu.cn)

Abstract: A particle swarm optimization algorithm based on multi-species cooperative evolution is presented. In this approach, the result space separation based on immune system is introduced into the particle swarm optimization (PSO) initial step, which reinforces the stability and global exploration ability of the PSO algorithm. In the evolution process, the best result value space is generated by using the multi-species evolution information, which is explored to induce the new particle swarm generated by stochastic mutation operation to fly into the better result space, which can avoid the premature convergence and speed up the convergence. The comparison of the performance of the proposed approach with that of traditional PSO algorithm and other multi-species cooperative particle swarm optimization algorithms is experimented. The experimental results show that the proposed method can not only effectively solve the premature convergence problem, but also significantly speed up the convergence and improve the stability.

Key words: Particle swarm optimization algorithm; Multi-species cooperative evolution; Immune system; The best result value space

1 引 言

微粒群算法(PSO)是基于群体智能理论的优化算法, 通过群体中微粒间合作与竞争产生的群体智能指导优化搜索[1-4]. 由于其结构简单、易于实现, 在函数优化和神经网络权值训练等方面都有很好的表现. 然而, 在进化过程中, PSO 受当前最优位置的影响, 在求解多峰函数优化问题时会出现早熟收敛和收敛速度慢等缺点[5-8]. 为了解决这一问题, Bergh 等[9]提出了一种协同进化微粒群算法(CPSO), 并在多峰函数优化的仿真实验中得到了比标准 PSO 更加满意的结果. 但 CPSO 对子群划分的依赖性较强, 不同的划分方法对算法性能的影响较大. 对此, 文献[10, 11]借鉴梯阶进化思想提出多种群协同进化微粒群算法(MCPSO), 并在神经网络结构进化设计中得到了很好的应用. 然而算法中并没有明确给出子群的划分方法, 同时子群间的信息没有共享, 协同操作体现的并不明显, 因此大大影响该算法的全局解收敛性能.

鉴于此, 本文提出一种改进的 MCPSO. 该算法利用阴性免疫算法实现解空间中子群的划分, 最大

收稿日期: 2008-11-22; 修回日期: 2009-04-21.
基金项目: 黑龙江省博士后基金项目(3236301199); 哈尔滨工程大学校科研基金项目(002080260735).
作者简介: 陶新民(1973—), 男, 安徽蚌埠人, 副教授, 博士, 从事智能信号处理、软计算的研究; 徐晶(1974—), 女, 黑龙江鸡西人, 博士, 从事进化计算、小波信号处理的研究.

限度地实现空间覆盖, 增加了子群的多样性, 全局最优解的搜索能力大大提高. 同时, 本文借鉴文化算法[12]的思想, 在子群进化后期对超级个体优胜区域进行更新, 引入变异操作生成新的种群, 增加了种群的多样性, 在增强算法全局解搜索能力的同时加快了收敛速度. 利用实验对不同实际数据进行测试均显示出改进算法的优良性能.

2　改进的 MCPSO

2.1　基本微粒群优化算法

微粒群优化算法中, 每个微粒代表问题的一个可能解, 它具有位置、速度两个特征. 微粒位置坐标对应的目标函数值即可作为该微粒的适应度. 算法通过适应度来评价微粒的优劣. 首先初始化一群随机微粒, 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: 一个是粒子本身所找到的最优解, 即个体极值 p_{Best}; 另一个是目前整个微粒群找到的最优解, 称为全局极值 g_{Best}. 微粒找到上述两个极值后, 根据下式来更新自己的速度与位置:

$$v_i(n+1) = wv_i(n) + c_1\,\mathrm{rand}_1()(p_{\mathrm{Best}} - p_i(n)) + c_2\,\mathrm{rand}_2()(g_{\mathrm{Best}} - p_i(n)), \quad (1)$$

$$p_i(n+1) = p_i(n) + v_i(n+1). \quad (2)$$

其中: $v_i(n)$ 是当前粒子的速度; $p_i(n)$ 是粒子的当前位置; rand() 是[0, 1] 之间的随机数; c_1 和 c_2 为学习因子, 通常 $c_1 = c_2 = 1$; w 是加权系数, 一般在0.1至0.9之间取值. 文献[2] 通过大量实验表明, 如果 w 随算法迭代的进行而线性减小, 则将显著改善算法的收敛性能. 设 $w_{\max}$ 为最大加权系数, $w_{\min}$ 为最小加权系数, run 为当前迭代次数, runMax 为算法迭代总次数, 则有

$$w = w_{\max} - \mathrm{run}\,\frac{(w_{\max} - w_{\min})}{\mathrm{runMax}}. \quad (3)$$

在更新过程中, 微粒每一维的最大速率不超过 $v_{\max}$, 坐标也限制在允许范围之内. 这样, p_{Best} 与 g_{Best} 在迭代过程中不断更新, 最后输出的 g_{Best} 就是算法得到的最优解.

2.2　MCPSO

多种群协同进化算法[10] 改变了传统的利用一个种群在解空间搜索最优解的方式, 将整个种群划分为几个子群, 每个子群代表问题的一个子目标, 所有子群在独立进化的同时, 基于信息迁移与知识共享共同进化. 因为子群进化相比微粒进化而言要漫长得多, 所以其进化周期 T 用个体层进化代数的正整数倍表示. 当群体适应度方差[7] 小于一定的阈值时, 该种群为成熟子群; 反之, 则为成长子群. 多种群协同进化的信息迁移与知识共享是通过超级个体集合实现的. 当子群达到成熟时, 子群中的最优个体作为超级个体进入超级个体集合, 若超级个体集合容量已达到设定值, 则用新进入的超级个体代替原有的适应度最低的超级个体; 然后重新对超级个体的适应度进行排序, 取适应度最好的超级个体为问题的最优解[11].

2.3　改进的 MCPSO

由于子群达到成熟时, 最优个体进入超级个体集合后不再继续进化, 这使得超级个体集合的信息没有共享. 为了增强种群的多样性, 充分利用各子群的进化信息, 加快算法的收敛速度, 在 MCPSO 算法中引入变异操作. 设种群 $P_j(j = 1, 2, \ldots, k$, 其中 k 是预定的种群数) 中的微粒 X_i 表示为

$$X_i = \{\rho_j, x_{i1}, x_{i2}, \ldots, x_{in}\}.$$

其中: ρ_j 表示子群的位置特征分量, x_{in} 表示微粒在子群中的自由位置分量. 设

$$I_{ij} = [l_{ij}, \mu_{ij}] = \{x_{ij} \mid l_{ij} \leqslant x_{ij} \leqslant \mu_{ij},\ x_{ij} \in R\}$$

表示第 i 个微粒的第 j 个分量的优胜区间, l_{ij} 和 μ_{ij} 分别表示该分量优胜区间的下限和上限. $\langle I_j, L_j, U_j\rangle (j = 1, 2, \ldots, n)$ 表示第 j 个分量的优胜区域, L_j 和 U_j 分别为该 j 个分量优胜区间的下限和上限所对应的适应值. 假设超级个体集合中第 i 个超级个体作用于第 j 个分量的下限, 第 i' 个超级个体作用于第 j 个分量的上限, 则更新规则如下:

$$l_j^{t+1} = \begin{cases} x_{i,j}^t, & \text{if } x_{i,j}^t \leqslant l_j^t \text{ or } \mathrm{obj}(x^t) < L_j^t; \\ l_j^t, & \text{otherwise.} \end{cases} \quad (4)$$

$$L_j^{t+1} = \begin{cases} \mathrm{obj}(x_i^t), & \text{if } x_{i,j}^t \leqslant l_j^t \text{ or } \mathrm{obj}(x_i^t) < L_j^t; \\ L_j^t, & \text{otherwise.} \end{cases} \quad (5)$$

$$\mu_j^{t+1} = \begin{cases} x_{i',j}^t, & \text{if } x_{i',j}^t \geqslant \mu_j^t \text{ or } \mathrm{obj}(x_{i'}^t) > U_j^t; \\ \mu_j^t, & \text{otherwise.} \end{cases} \quad (6)$$

$$U_j^{t+1} = \begin{cases} \mathrm{obj}(x_{i'}^t), & \text{if } x_{i',j}^t \geqslant \mu_j^t \text{ or } \mathrm{obj}(x_{i'}^t) > U_j^t; \\ U_j^t, & \text{otherwise.} \end{cases} \quad (7)$$

其中: l_j^t 和 μ_j^t 表示超级个体集合中第 t 代第 j 个分量的下限和上限, L_j^t 和 U_j^t 分别表示其适应度, $x_{i,j}^t$ 表示第 t 代第 i 个个体的第 j 个分量, obj() 为目标函数. 当子群成熟时, 利用超级个体优胜区域的知识进行变异, 在个体优胜区域内随机生成粒子代替成熟种群的粒子重新进行进化, 即

$$x_{ij}^{t+1} = l_j^t + (\mu_j^t - l_j^t)\,\mathrm{rand}(\bullet), \quad i = 1, 2, \ldots, n, \quad (8)$$

其中 rand(·) 为 0 ~ 1 的随机数.

通过引入超级个体优胜区域, 在优胜区域内随机生成粒子代替成熟种群的粒子重新进行进化, 可以进一步探索全局解空间, 使得子群向着最优解的区域逼近, 加快算法的收敛速度, 在增强全局解搜索能力的同时提高了算法收敛速度.

2.4 子微粒群的划分方法

为了能最大限度地实现子群对整个空间的覆盖, 本文提出一种新的基于免疫原理的子微粒群划分方法. 该方法借鉴免疫系统中免疫抑制原理, 对相近的两个粒子进行移动, 实现最大解空间的覆盖. 算法描述如下:

$$d = (c, r_d),\ c = (c_1, c_2, \cdots, c_n),\ r_d \in R,$$

其中 c 代表检测器中心点的坐标, 也是解空间微粒的位置. 亲和力的度量公式为

$$D(x, y) = \left(\sum |x_i - y_i|^{\lambda}\right)^{\frac{1}{\lambda}},$$

$$x = (x_1, x_2, \cdots, x_n),\ y = (y_1, y_2, \cdots, y_n). \quad (9)$$

$\lambda = 2$ 时即为欧氏距离, 距离小则亲和力高. 这种方法允许寻找具有代表性的检测器覆盖解空间. 若检测器 $d = (c, r_d)$ 与解空间中最近微粒的距离是 D, 则该检测器的半径 $r_d = D/2$.

算法流程如下(num_iter 为循环次数):

Step1: 初始化 k 个粒子位置.

Step2: 当循环次数 $i <$ num_iter, 不满足停止条件时循环.

Step3: 对于每个微粒 c, 计算最近邻微粒 c^{nst} 并进行移动, 依据下式移动检测器:

$$c^{new} = c \pm \alpha \frac{\mathrm{dir}}{\|\mathrm{dir}\|},\ \mathrm{dir} = c - c^{nst}, \quad (10)$$

$$\alpha = F e^{-i/p}. \quad (11)$$

其中: α 随着循环次数 i 的增加而递减, F 控制移动步长, p 为调节因子.

Step4: 计算 $r_d = \|c - c^{nst}\|/2$.

Step5: 对每一维自变量 $j \in n$ 进行规范化处理, 如果 $\|c_j + r_d\| > L_j$, 则 $c_j^{new} = L_j - r_d$; 如果 $\|c_j - r_d\| < S_j$, 则 $c_j^{new} = S_j + r_d$, 最终生成新的检测器 $d = (c^{new}, r_d)$. L_j 为第 j 分量的最大值, S_j 为第 j 分量的最小值.

Step6: 循环结束, 生成的检测器中心 c 的坐标即是各个子微粒群的中心. 以 c 为中心, 为了最大限度地实现空间的覆盖, 以 $3r_d/2$ 为半径随机生成 m 个粒子, 即

$$P_l X_i = c_l \pm \mathrm{rand}() \times 3r_l/2,$$

$$i = 1, 2, \cdots, m,\ l = 1, 2, \cdots, k, \quad (12)$$

其中 $\pm$ 根据新的 rand() 随机确定, 大于 0.5, 取 +; 否则, 取 -.

本文以二维空间 $[0, 1]$ 范围生成 4 个粒子来说明基于免疫原理的子微粒群划分方法, 具体效果如图 1 所示.

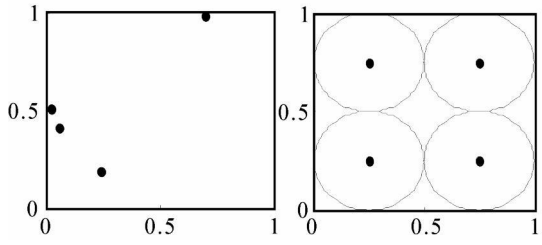

图 1 基于免疫原理的子微粒群划分方法示例图

2.5 改进的 MCPSO 算法流程

设 θ 为种群 j 的位置特征分量, $P_j X_i$ 为种群 P_j 中第 i 个微粒目前的自由位置矢量, $P_j Y_i$ 为该微粒目前为止经过的最好位置, $P_j Y$ 为种群 P_j 中所有微粒到目前为止经历过的最好位置. P_j 中的第 i 个微粒可表示为 $b(\theta_j, P_j X_i)$, 其适应度函数为 $f(b(\theta_j, P_j X_i))$, 则改进的多种群协同进化微粒群算法流程表述如下:

Step1: 利用基于免疫原理的子微粒群划分方法初始化产生 k 个子群, 即

$$b(\theta_j, P_j X_1, P_j X_2, \cdots, P_j X_m),$$

$$j = 1, 2, \cdots, k,\ i = 1, 2, \cdots, m.$$

对于每个子群 $j \in (1, 2, \cdots, k)$ 分别进行迭代.

Step2: 对于每个子群 j 中的粒子 $i \in (1, 2, \cdots, m)$, 如果 $f(b(\theta_j, P_j X_i)) < f(b(\theta_j, P_j Y_i))$, 则 $P_j Y_i = P_j X_i$; 如果 $f(b(\theta_j, P_j X_i)) < f(b(\theta_j, P_j Y))$, 则 $P_j Y = P_j X_i$.

Step3: 利用式(1)和(2), 更新微粒 $b(\theta_j, P_j X_i)$.

Step4: 如果 $\mathrm{mod}(t, T) = 0$, 则进入种群进化阶段(5), 否则执行式(9).

Step5: 如果 $\sigma^2(t) < \varepsilon$, 则说明该种群进入收敛状态, 将 $P_j Y$ 放入超级个体集合.

Step6: 超级个体按照适应度重新排序.

Step7: 按照式(4) ~ (7) 更新超级个体优胜区域.

Step8: 利用式(8) 实施变异操作生成新种群 θ_j, 重新进行进化.

Step9: 符合算法停止条件, 输出超级个体集合中的最优个体, 作为问题的最优解.

3 实验分析

3.1 改进 MCPSO 的参数设置及 Benchmark 函数

为了评价改进 MCPSO 的全局搜索性能、收敛速度和算法的稳定性, 选择 CPSO, MCPSO 和改进

的 MCPSO 进行对比实验. 传统的理论分析显示, 线性下降的惯性权重有利于种群搜索, 所以, 选择 w 在$[0.95, 0.4]$ 之间随迭代次数线性递减, c_1 和 c_2 均为 1. 为了排除算法内部随机操作对算法性能的影响, 以 1000 次实验的统计结果进行分析. 子微粒群划分算法中参数设置步长 F 为 0.5, 调节因子 p 为 6, 循环3000次. 所有实验结果均在相同环境下通过 Matlab 编程计算得出.

选择 PSO 和遗传算法经常使用的 4 个 Benchmark 函数问题进行实验, 并根据函数性质分为具有单一极小点(单模态) 和多个局部极小点(多模态) 两大类. 单模态选择Rosenbrock 函数, 它是一个经典的复杂优化问题, 其全局最优点位于一个平滑、狭长的抛物线形山谷内, 因为函数仅为优化算法提供了少量信息, 使算法很难辨别搜索方向, 找到全局最小点的机会较小, 所以 Rosenbrock 函数通常用来评价优化算法的执行效率. 多模态函数选择了 3 个经典的非线性多模态函数, 它们具有广泛的搜索空间、大量的局部极小点和高大的障碍物, 通常被认为是很难处理的复杂多模态问题. 下列公式给出了这 4 个函数的定义、取值范围和全局最优解:

1) Rosenbrock 函数

$$f_1 = \sum_{i=1}^{n}(100(x_{i+1} - x_i^2) + (x_i - 1)^2), \quad (13)$$

取值范围为$[-15, 30]$, 维度为 4, 最小值和位置为 $0(1, \cdots, 1)$.

2) Rastrigin 函数

$$f_2 = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10), \quad (14)$$

取值范围为$[-15, 30]$, 维度为 8, 最小值和位置为 $0(0, \cdots, 0)$.

3) Griewank 函数

$$f_3 = \frac{1}{4000}\sum_{i=1}^{n}(x_i)^2 - \prod_{i=1}^{n}\cos((x_i/\sqrt{i})) + 1, \quad (15)$$

取值范围为$[-15, 30]$, 维度为 10, 最小值和位置为 $0(0, \cdots, 0)$.

4) Ackley 函数

$$f_4 = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e, \quad (16)$$

取值范围为$[-15, 30]$, 维度为 15, 最小值和位置为 $0(0, \cdots, 0)$.

3.2　单模态 Benchmark 问题对比实验

CPSO, MCPSO 及改进 MCPSO 采用 2 个子群进行优化, 微粒的数目为 20 个, 每个子群的微粒数为 10 个, 循环次数为 2000 次, 种群进化周期 T 为 80, 群体适应度方差阈值为 0.06. 实验结果如图 2 所示.

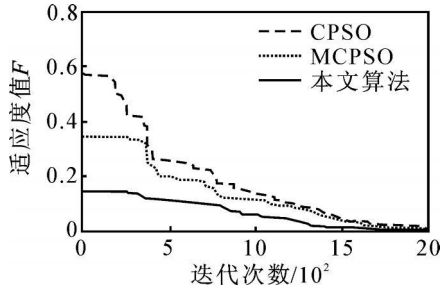

图 2　4 维 Rosenbrock 函数收敛性能对比

对于复杂单模态 4 维 Rosenbrock 函数问题, 本文提出的算法具有较快的收敛速度. 在算法的初始阶段, 由于 Rosenbrock 函数仅给算法提供了很少的信息, 使算法不能有效地辨别搜索方向, 图 2 中所有算法都相对稳定, 用来寻找搜索方向. 对于本文算法, 由于初始化时采用免疫原理的初始种群划分方法, 解空间覆盖率较高, 算法能很快找到较好的搜索方向, 并具有较快的收敛速度.

表 1 中 Rosenbrock 函数的均值、方差的数据显示, 本文算法全局最优解搜索能力大大提高, 这是由于本文算法增加了变异操作, 扩大了解空间的搜索范围, 使得搜索方向向着最优解子空间逼近. 同时可以看出, 由于 CPSO 和 MCPSO 在种群初始化中具有随机性, 使得算法的执行结果也具有随机性; 而本文算法由于初始子群解空间的稳定划分, 单次执行结果之间的相差明显小于其他算法, 具有较好的稳定性和健壮性. 为了考察不同微粒个数对算法收敛性能的影响, 对 32 个微粒实验的结果进行对比, 实验表明收敛性能提高并不受微粒个数的影响.

表 1　3 种算法在 Rosenbrock 问题上的收敛性能比较

算法	粒子数	均值	方差
CPSO	20	1.6189E-004	2.0539E-004
	32	2.3162E-007	3.5942E-007
MCPSO	20	9.2502E-005	1.0457E-004
	32	3.5669E-008	4.3760E-008
本文算法	20	6.1511E-006	9.7830E-006
	32	6.0770E-010	2.4731E-010

3.3　多模态 Benchmark 问题对比实验

对于多模态 Rastrigrin 函数, CPSO, MCPSO 及改进 MCPSO 的子群个数为 2, 微粒个数为 20, 微粒的维度为8, 迭代次数为8000代, 种群进化周期 T 为 100, 群体适应度方差阈值为 0.06. 实验结果如图 3 所示.

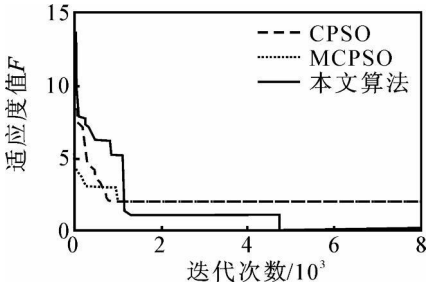

图 3 8 维 **Rastrigrin** 函数收敛性能对比

多模态 Rastrigrin 函数是一个经典的用来测试全局搜索性能的函数. 图 3 表明, CPSO 和 MCPSO 在搜索后期容易陷入局部极小点, 不能进一步找到全局最小点. 实验中局部极小点的适应度为 1.9899 和 2.9849; 而本文算法能持续寻找全局最小点, 并且找到近似 0 的全局最优解. 表 2 表明, 虽然不是每一次运行都能找到近似全局最优解 0, 但是只要给予足够长的迭代次数, 本文算法总能逃出局部极小点找到全局最小点. 因此, 本文算法具有良好的全局搜索性能和较快的搜索速度. 同样, 为了考察粒子个数对收敛性能的影响, 取 40 个粒子进行实验, 结果表明对于多模态 Rastrigrin 问题, 本文算法收敛性能的提高仍不受粒子个数的影响.

表 2 3 种算法在 **Rastrigrin** 问题上的收敛性能比较

算法	粒子数	均值	方差
CPSO 算法	20	2.3216	1.6248
	40	1.9899	0.6293
MCPSO	20	2.1190	0.7490
	40	1.1608	0.7356
本文算法	20	0.2291	0.4062
	40	0.1960	0.2324

对于 10 维的多模态 Griewank 函数, 子群个数为 2, 粒子个数为 40, 粒子的维度为 10, 迭代次数为 10000 代, 种群进化周期 T 为 200, 群体适应度方差阈值为0.06. 实验结果如图 4 所示.

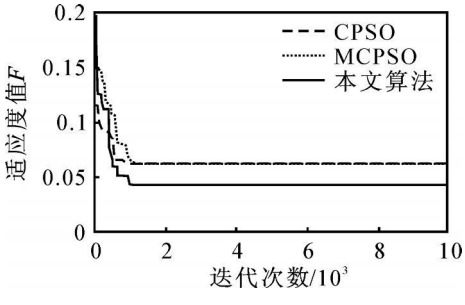

图 4 10 维 **Griewank** 函数收敛性能对比

对于 10 维的多模态 Griewank 问题, 本文算法在 1000 次实验中有 26% 的机会找到近似全局最优解 0, 而 CPSO 和 MCPSO 找到近似全局最优解 0 的概率分别为 6% 和 13% 左右, 全局搜索能力显著优于 CPSO 和 MCPSO, 能够有效逃出局部极小点找到全局最小点. 本文算法的均值和方差数据均优于其他算法, 显示了新算法的稳定性和健壮性. 但是, 实验中发现本文算法并不能保证在有限次迭代内即可找到全局极小点. 表 3 的数据显示本文算法也具有陷入局部极小的可能, 这是由于 Griewank 函数各维之间显著相关, 某一维位置的变化并不能提高微粒的适应度, 只有当各维同时发生变化时才能提高微粒的适应度, 这使得微粒逃出极小点的概率非常小, 图 4 中算法后期阶段的平坦性恰恰说明了这一点.

表 3 3 种算法在 **Griewank** 问题上的性能比较

算法	粒子数	均值	方差
CPSO	40	0.1345	0.0376
	60	0.1150	0.0438
MCPSO	40	0.0752	0.0256
	60	0.0650	0.0322
本文算法	40	0.0413	0.0086
	60	0.0066	0.0032

图 5 显示了 15 维 Ackley 函数的运行结果, 参数设置为 2 个子群, 40 个粒子, 迭代次数为 10000 代, 种群进化周期 T 为 200, 群体适应度方差阈值为 0.06.

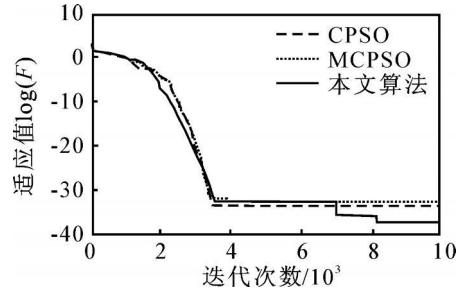

图 5 15 维 **Ackley** 函数收敛性能对比

从图 5 可以看出, 3 种算法在搜索的初始阶段都具有较快的收敛速度, 能保持正确的搜索方向. 然而到了中后期, CPSO 和 MCPSO 算法陷入了局部极小点, 无法继续寻找全局最小点. 而本文算法由于变异操作的引入, 性能显著优于其他算法, 但与其他算法一样, 需要一个缓慢的过程才能跳出局部极小点. 这与 Griewank 函数一样, 是由于微粒没有充分利用各维之间的信息所导致的. 表 4 显示了本文算法同样具有良好的稳定性和健壮性. 多模态 Benchmark 问题的实验结果显示, 由于变异操作的引入, 种群的全局搜索能力伴随着搜索的全部过程, 同时最优优胜区域的指导使得变异粒子能够很好地向最优解空间范围逼近, 并不会随着迭代次数的增加而消失. 表 2 ~ 表4的数据还表明, 本文算法由于采用基于免疫

表 4　3 种算法在 Ackley 问题上的收敛性能比较

算法	粒子数	均值	方差
CPSO	40	3.8487E-015	1.8346E-015
	60	2.6645E-015	1.2661E-015
MCPSO	40	5.2172E-015	1.0278E-015
	60	3.1169E-016	8.4529E-016
本文算法	40	6.5722E-017	1.1873E-018
	60	1.3257E-018	8.9973E-019

原理的子群划分方法, 具有很强的稳定性, 实验结果的差异很小, 而其他两种算法都不稳定, 这是由于算法在初始化阶段微粒分布不均匀造成的.

4　结　　论

本文提出了一种改进的基于多种群协同进化的微粒群优化算法. 结合实验得到如下结论:

1) 借鉴文化算法的思想, 新算法利用种群进化信息生成解的优胜区域, 找出更有效的最优解子空间, 通过变异操作使得微粒向着最优解空间逼近, 使其在进化过程中始终保持最优解空间勘探的能力, 增强了算法全局解搜索能力.

2) 通过基于免疫原理的初始子群划分方法, 使初始化时微粒的分布更加均匀, 提高了算法的稳定性和全局解搜索能力.

3) 利用 Benchmark 函数, 对 CPSO 算法、MCPSO 算法和本文算法的全局解搜索能力以及收敛速度进行对比. 实验结果表明本文算法有效提高了全局解搜索能力, 算法比较稳定.

需要指出的是, 对于子群个数对全局解空间搜索能力的影响, 本文算法没有进行讨论, 这将是本课题下一步研究的重点.

参考文献(References)

[1] Kennedy J. The particle swarm: Social adaptation of knowledge[C]. Proc of IEEE Int Conf on Evolutionary Computation. Piscataway NJ: IEEE Press, 1997: 303-308.

[2] 马慧民, 叶春明. 基于文化进化的并行粒子群算法[J]. 计算机工程, 2008, 34(2): 193-195.
(Ma H M, Ye C M. Parallel particle swarm optimization algorithm based on cultural evolution[J]. Conputer Engineering, 2008, 34(2): 193-195.)

[3] Daneshyari M, Yen G G. Cultural MOPSO: A cultural framework to adapt parameters of multiobjective particle swarm optimization [C]. 2008 IEEE Congress on Evolutionary Computation. Hong Kong, 2008: 1325-1332.

[4] Hou Y L, Hu J W, Zhao C H, et al. Test set optimization using cultural particle swarm optimization [J]. J of Harbin Engineering University, 2007, 27(7): 151-154.

[5] Liu S, Wang X. Cultured differential particle swarm optimization for numerical optimization problems [C]. The 3rd Int Conf on Natural Computation. Haikou, 2007: 642-646.

[6] 彭宇, 彭喜元, 刘兆庆. 微粒子算法参数效能的统计分析[J]. 电子学报, 2004, 32(2): 209-213.
(Peng Y, Peng X Y, Liu Z Q. Statistic analysis on parameter efficiency of particle swarm optimization[J]. Acta Electronica Sinica, 2004, 32(2): 209-213.)

[7] 吕振肃, 侯志荣. 自适应变异的粒子群优化算法[J]. 电子学报, 2004, 32(3): 416-420.
(Lv Z S, Hou Z R. Particle swarm optimization with adaptive mutation[J]. Acta Electronica Sinica, 2004, 32(3): 416-420.)

[8] 赫然, 王永吉. 一种改进的自适应逃逸微粒群算法及实验分析[J]. 软件学报, 2005, 16(12): 2036-2044.
(He R, Wang Y J. An improved particle swarm optimization based on self-adaptive escape velocity[J]. J of Software, 2005, 16(12): 2036-2044.)

[9] Bergh F, Engelbrecht A P. A cooperative approach to particle swarm optimization [J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 1-15.

[10] 王俊年, 申群太. 基于多种群协同进化微粒群算法的径向基神经网络设计[J]. 控制理论与应用, 2006, 32(2): 251-255.
(Wang J N, Shen Q T. Evolutionary design of RBF neural network based on mulit-species cooperative particle swarm optimizer[J]. Control Theory and Applications, 2006, 32(2): 251-255.)

[11] 王俊年, 申群太. 基于种群小生境微粒群算法的前向神经网络设计[J]. 控制与决策, 2005, 20(9): 981-985.
(Wang J N, Shen Q T. Evolutionary design of feed-forward neural network based on sepecies niching particle swarm optimizer[J]. Control and Decision, 2005, 20(9): 981-985.)

[12] 黄海燕, 顾幸生. 基于文化算法的神经网络及其在建模中的应用[J]. 控制与决策, 2008, 23(4): 477-480.
(Huang H Y, Gu X S. Neural network based on cultural algorithms and its application on modeling[J]. Control and Decision, 2008, 23(4): 477-480.)