文章编号: 1001-0920 (2011) 05-0700-07

基于克隆多尺度协同开采的离散微粒群算法

陶新民[1], 徐 晶[2], 王 妍[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. 黑龙江省科技学院 数力系，哈尔滨 150027)

摘 要: 提出一种克隆多尺度协同开采的离散微粒群算法. 多尺度变异概率根据粒子适应值大小进行动态调节, 在算法初期通过大尺度概率变异增加算法多样性, 后期通过逐渐减小的小尺度变异提高算法在最优解附近的局部精确解搜索性能, 对当前最优解进行克隆选择, 可进一步增强算法逃出局部极小解的能力以及所求解的精度. 将算法应用于 5 个 benchmark 函数优化问题并与其他算法比较, 结果表明该算法不仅能增强全局解搜索性能, 同时最优解的精度也有所提高.

关键词: 离散微粒群；克隆；多尺度；协同开采

中图分类号: TP18 **文献标识码:** A

Discrete particle swarm optimization based on clone multi-scale cooperative exploitation

TAO Xin-min[1], *XU Jing*[2], *WANG Yang*[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 discrete particle swarm optimization(DPSO) algorithm based on multi-scale cooperative clone mutation (MSCMDPSO) is proposed. The clone mutation operator with multi-scale possibilities is introduced on the current optimical solution, which can not only improve the ability of local search, but also keep the abilities of global space search and escaping from local optima. The mutation operator with large-scale possibilities can be utilized to quickly localize the global optimized space at the early evolution. The scale-changing strategy produces a smaller multi-scale mutation operators according to the variation of the fitness value and makes mutation operators with smaller-scale possibilities implement local accurate minima solution search at the late evolution. The experiment studies on 5 standard benchmark functions, and the experimental results show the proposed method can not only effectively solve problem of lack of local search ability, but also significantly speed up the convergence and improve the stability.

Key words: discrete particle swarm optimization；clone；multi-scale；cooperative exploitation

1 引 言

微粒群优化 (PSO) 是 Kennedy 和 Eberhart 于 1995 年提出的一种新兴的基于群体智能理论的演化计算技术. 它源于对鸟群觅食运动行为的模拟, 相对于遗传算法而言, PSO 具有运算简单、易于实现、需要调节的参数少以及较强的全局收敛能力和鲁棒性好等优势. 目前已广泛应用于科学和工程领域, 如函数优化、神经网络训练和模糊分类、系统控制等[1-2]. PSO 最初是为解决非线性连续优化问题而设计的, 具有连续本质. 为了将其应用于解决离散问题, Kennedy 于 1997年提出了解决 0-1 规划问题的离散微粒群优化算法(DPSO), 这里称之为传统离散微粒群算法[3-5].

由于传统离散微粒群算法同样具有收敛精度低、易陷入局部最优解等问题[6-7], 国内外学者提出了多种算法来改进传统离散微粒群算法的优化性能, 一方面是将对连续微粒群算法的改进直接应用到离散微粒群算法中. 例如: 1) 设置自适应的控制参数来改进离散 PSO 的性能. 如文献 [8] 采用惯性权重线性下

收稿日期: 2010-02-01；**修回日期:** 2010-05-21.

基金项目: 国家自然科学基金项目(61074076)；中国博士后科学基金项目(20090450119)；中国博士点新教师基金项目(20092304120017)；黑龙江省博士后基金项目(LBH-Z08227).

作者简介: 陶新民(1973−), 男, 副教授, 从事智能信号处理、智能计算等研究；徐晶(1974−), 女, 副教授, 从事模式识别的研究.

降法, 使算法在搜索初期有效地探索较大区域, 随着搜索过程的深入, 逐渐开始精细搜索. [9] 则设计了自适应的最大速度阈值参数, 使粒子在搜索过程初期倾向于全局搜索, 在算法后期倾向于当前最优解附近的局部搜索, 有效地实现了粒子全局和局部搜索性能的平衡 (VCDPSO). 2) 增强粒子局部极值的逃逸能力. 如 [10] 通过对惯性因子采用自适应扰动机制来增强微粒群跳出局部最优的能力 (DDPSO). 3) 增加微粒群多样性, 摆脱群体陷入局部极值的可能, 提高算法性能[11-12]. 上述改进算法没有考虑到 DPSO 算法的进化过程与连续PSO算法截然不同这一重要区别, 直接将对连续算法的改进应用到离散微粒群算法中, 因此上述改进算法并不适用于离散微粒群算法, 没有真正达到弥补离散微粒群算法不足的目的[13]. 另一方面是直接对离散微粒群算法进行分析, 例如 [14] 分析了 DPSO 中微粒速度的变化, 提出一种双速度状态跟踪的 DPSO 算法 (简称 NDPSO), 消除了原有算法对参数敏感的不足, 但由于算法增加了微粒状态改变的随机性, 其全局优化性能同样受到影响. [15] 对 DSPO 算法微粒状态转移情况进行了分析, 提出一种改进的离散化方法 (MDPSO), 降低了微粒在最优解附近发散的可能, 然而该算法在降低微粒发散度的同时, 也使得其易陷入局部最优解. 因此如何在提高算法局部搜索能力的同时不影响算法逃出局部最优解的能力值得深入研究.

本文在分析传统 DPSO 算法微粒状态转移情况的基础上, 提出一种克隆多尺度协同开采的离散微粒群算法 (MSCMDPSO). 该算法在克隆当前微粒的基础上进行多尺度概率变异, 并且变异概率算子随着适应值的提升而逐渐减少, 这样既可提高对解空间的开采能力, 又可保持算法的勘探能力, 使其能有效逃出局部最优解, 保证了算法的收敛性能. 将该算法应用到各种不同优化函数并与其他改进的 DPSO 算法进行比较, 试验结果表明本文算法具有优良的优化性能.

2　离散微粒群算法描述

连续微粒群优化算法的基本思想是模拟鸟群捕食的行为, 将优化问题的每个潜在解看作搜索空间的一个"微粒", 所有的微粒都由一个适应函数来决定其适应值. 此外, 每个微粒还有一个速度决定它的运动方向和移动距离. PSO 算法初始化一群随机微粒 (随机解), 然后通过迭代找到最优解. 在每次迭代中, 微粒通过跟踪 2 个"极值"来更新自己: 1) 微粒本身所找到的最优解, 这个解称为个体极值; 2) 整个种群或者该微粒的某个邻域范围目前找到的最优解. 连续算法的机制适合于求解无约束的连续型最优化问题, 并且取得了良好效果. 为拓展微粒群优化算法的应用范围, Kennedy 和 Eberhart 对连续微粒群算法进行修正, 提出一个二进制离散微粒群算法[3], 用于处理离散型优化问题.

设微粒群含有 L 个微粒, 每个微粒相当于 N 维离散空间的一个活动点. 微粒 i 在时刻 t 的速度、位置、个体最好位置和全局最好位置分别用 $\boldsymbol{v}_i(t)$, $\boldsymbol{x}_i(t)$, $\boldsymbol{x}_i^{(p)}(t)$, $\boldsymbol{x}_i^{(g)}(t)$ 表示, 则微粒 i 的速度和位置的各维分量迭代公式如下[3]:

$$\begin{aligned}v_{ij}(t+1) =& wv_{ij}(t) + c_1 r_{1j}(t)(x_{ij}^{(p)}(t) - x_{ij}(t)) + \\ & c_2 r_{2j}(t)(x_{ij}^{(g)}(t) - x_{ij}^{(t)});\end{aligned} \tag{1}$$

$$x_{ij}(t+1) = \begin{cases} 0, & \rho \geqslant \mathrm{Sig}(v_{ij}(t+1)); \\ 1, & \rho < \mathrm{Sig}(v_{ij}(t+1)). \end{cases} \tag{2}$$

其中: $i = 1, 2, \cdots, L$; $j \in \{1, 2, \cdots, N\}$ 表示微粒编码中分量的维数; $r_{1j}(t)$ 和 $r_{2j}(t)$ 是 (0,1) 上均匀分布的随机数; w, c_1 和 c_2 是权重及加速系数; $\rho \sim U[0, 1]$ 是 (0,1) 区间上均匀分布的随机变量; Sig( ) 表示 sigmoid 函数, 本文取[3] $\mathrm{Sig}(x) = \dfrac{1}{1+\exp(-x)}$.

虽然二进制离散微粒群优化算法的速度迭代公式与连续微粒群优化算法相同, 但限制速度的参数 $v_{\max}$ 及加速系数 w, c_1, c_2 的意义与其在连续微粒群优化算法中的意义完全不同. 文献 [3] 规定微粒的最大速度 $v_{\max}$ 满足 $|v_{\max}| = 4$, 这使得 $0.018 \leqslant \mathrm{Sig}(v_{ij}(t+1)) \leqslant 0.982$.

由文献 [14-15] 对微粒运动轨迹的分析可知, 在个体极值和全局极值之间不相等时, 微粒会在两个极值间来回震荡, 当 c_1 和 c_2 取值相等时, 会出现均匀震荡趋势, 此时粒子趋于随机搜索, 勘探能力较强, 直到两个极值趋于一致. 在两个极值相等且微粒不等于这两个极值的情况下, 微粒会呈现向极值收敛的趋势, 体现出一种有目的的寻优过程, 且逐渐向两个极值解逼近, 勘探能力同样较强. 当微粒收敛到全局最优解时, 如果 $w > 1$, 则微粒会随着迭代而保持原有的状态, 变异的可能性降低, 趋于陷入局部极值解; 如果 $w \leqslant 1$, 则随着迭代次数的增加, w 会使处于当前最优的微粒的速度趋于零, 使得当前微粒产生变异的概率趋于 0.5, 且很快发散出去, 无法在最优解附近进行深度的局部搜索. 因此有效控制算法中处于当前最优解的微粒搜索的发散性, 提高算法局部精确解的搜索能力, 同时保持局部最优解的逃逸能力是传统 DPSO 算法需要解决的首要问题.

3　克隆多尺度协同开采的离散微粒群算法

3.1　多尺度概率变异算子

增加变异操作能帮助算法逃出局部最优解, 但大的变异概率无法保证逃逸后的新位置位于全局最优

解的附近, 即新位置的适应值不一定优于现有的最优解. 这样通过单纯的大尺度概率变异无法得到更优的位置, 进而在重新迭代的过程中, 因迭代次数的限制可能使其无法收敛到全局最优解. 由PSO算法的产生机理可知, 在算法进化的后期, 问题的最优解往往存在于当前最优解周围, 因此有必要在最优解周围进行精确的局部最优解搜索. 小的变异概率有利于进行局部搜索, 即增强算法的开采能力, 但其又易陷入局部极值解. 因此如何设计变异概率是协调离散微粒群算法勘探和开采能力的关键. 本文采用多尺度变异概率算子, 进化初期采用大尺度概率变异使得算法在搜索空间中进行分散式搜索, 同时变异算子随着适应值的提升而逐渐减小, 使得算法在进化后期的局部搜索能力增强, 提高了最优解的精度, 保证了算法的收敛性能. 算法具体描述如下:

设尺度个数为 M, K 为迭代次数. 首先根据适应值的大小对种群中的微粒由小到大进行排序、组合, 生成 M 个子群, 每一个子群的微粒个数为 $P = N/M$, K 为当前迭代次数. 计算每一个子群的适应值

$$\mathrm{Fit}X_m^{(K)} = \sum_{i=1}^{P} f(x_i^m)/P,\ i = 1, 2, \cdots, M, \tag{3}$$

其中 f 是微粒的适应值函数. 初始化多尺度概率变异算子的初始值

$$\boldsymbol{p}^{(0)} = (p_1^{(0)}, p_2^{(0)}, \cdots, p_M^{(0)}). \tag{4}$$

初始值可随机设定为0~1间的随机数, 随着迭代次数的增加, 不同尺度概率变异算子之间相互竞争, 根据适应能力的不同而设置不同的变异能力. 第 m 个变异概率算子的值为

$$p_m^{(K)} = p_m^{(K-1)} \exp\Big(\frac{M \cdot \mathrm{Fit}X_m^{(K)} - \sum_{m=1}^{M} \mathrm{Fit}X_m^{(K)}}{\mathrm{Fit}X_{\max} - \mathrm{Fit}X_{\min}}\Big), \tag{5}$$

$$\mathrm{Fit}X_{\max} = \max_{1 \leqslant i \leqslant M} (\mathrm{Fit}X_i^{(K)}),$$

$$\mathrm{Fit}X_{\min} = \min_{1 \leqslant i \leqslant M} (\mathrm{Fit}X_i^{(K)}). \tag{6}$$

由于多尺度概率变异算子的进化是一个递归过程, 某一尺度变异概率算子的值可能很大, 因此对变异算子的标准差做如下规范: 如果 $p_i^{(k)} > T$, 则

$$p_i^{(k)} = \big|T - p_i^{(k)}\big|. \tag{7}$$

T 为变异概率算子的阈值, 这里设置为0.7, 重复式(7)直到满足 $p_i^{(k)} < T$. 当 $M = 5$ 时不同尺度概率变异算子随迭代次数的变化如图1所示, 通过本文的多尺度概率变异算子能实现变异概率空间的覆盖. 其中大尺度有利于实现解空间的粗搜索, 可以逃逸出局部极小值, 快速定位到最优解区域, 具有一定的空间勘探能力; 小尺度能在进化后期实现最优解附近的局部精确解深度搜索, 具有较强的开采能力.

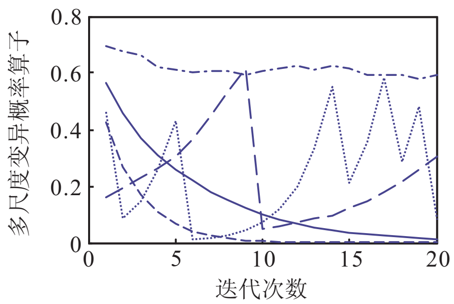

图1 不同尺度概率变异算子随迭代代数的变化示意图

3.2 克隆变异选择算法

为了进一步增强粒子在最优解区域的逃逸能力以及所求解的精度, 针对每个尺度概率变异, 克隆当前最优解, 对每一个克隆最优解个体, 产生随机向量. 如果某一维超过当前的变异概率, 则进行该维的状态取反操作, 实现变异. 比较所有变异尺度的克隆群体的适应值, 如果最好的适应值优于当前最优解, 则选择具有最好适应值的克隆变异个体为当前新的最优解.

3.3 惯性权重参数设定

在离散微粒群算法中, w 的取值在离散PSO算法中最难控制, 当 $|w| < 1$ 时, $v_{ij}^t \xrightarrow{t\to\infty} 0$, 而 $\mathrm{Sig}(0) = 0.5$, 使得微粒产生变异的概率趋近于0.5, 因此能有效防止早熟收敛, 但局部精确解搜索能力减弱; 当 $w > 1$ 时, 在算法初期, 速度会随着迭代而增加, 最终使得 $\mathrm{Sig}(v_{ij}) = 1$, 因此所有微粒位都将变成1; 当 $w < -1$ 时, 速度会随着迭代而减小, 最终使得 $\mathrm{Sig}(v_{ij}) = 0$, 微粒位变成0的概率增加. 因此文献[8-10]对惯性参数采用自适应或扰动机制来改善离散微粒群算法并不可取, 本文算法中取 $w = 1$ 实现折衷.

3.4 MSCMDPSO 算法流程

1) 算法的参数初始化设定;

2) 按照离散微粒群进化公式(1)进化;

3) 根据式(2)进行微粒状态的计算;

4) 计算当前最优解及每个微粒的个体极值解;

5) 根据式(3)~(7)计算多尺度概率变异算子值;

6) 克隆变异选择算法;

7) 循环直到终止条件.

4 克隆多尺度协同开采的离散微粒群算法

为了更有效地说明算法优化的机理, 设优化函数 $f(x)$ 只有一个自变量函数, 如图2所示. 本文算法中种群的某一个微粒经过多次迭代后到达 a 点, 经过小尺度概率克隆变异深度搜索后, 向下“下山”找到个体 a', 经过DPSO算法的进化, 进化到 a''; 然后经

过大尺度概率克隆变异操作后找到适应值更高的微粒 b, 则微粒 b 被选入下一世代. 经过 DPSO 若干代进化以及小尺度概率克隆变异操作后, 找到微粒 b'. 微粒 b' 经过大尺度概率克隆变异操作找到了微粒 c, 则 c 被选入下一个世代, 经过 DPSO 进化和小尺度概率克隆变异操作最终找到最优解 c'.

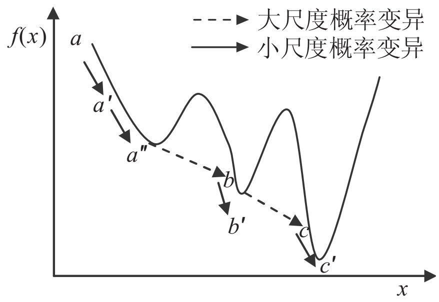

图 2　多尺度概率克隆变异的优化机理

由上可知, 多尺度概率克隆变异操作将整个解空间看作一个山谷, 经过大尺度概率克隆变异操作进行"下山"以寻求适应值更高的区域 ($a'' \to b, b' \to c$). 大尺度概率克隆变异操作用于搜索全局解空间, 是粗搜索. 若大尺度概率克隆变异操作找到了适应值更高的区域, 则 DPSO 进化操作和小尺度概率克隆变异操作能通过小区域搜索在该区域进行局部深度搜索以寻求更高精度的解 ($a \to a', a' \to a'', b \to b', c \to c'$), DPSO 进化操作和小尺度概率克隆变异操作用于搜索局部解空间. 因此本文算法是一个勘探和开采能力相互交替进行的搜索算法, 具有全局和局部两层邻域搜索机制, 从而保证算法全局和精确的局部寻优性能.

5　对比实验结果及分析

5.1　测试数据及实验设计

为分析本文提出的新算法的全局搜索性能、收敛速度和算法的稳定性, 本文选择 DPSO 和其改进的离散微粒群算法 DDPSO, NDPSO[14], MDPSO[15] 及 VCDPSO[9] 对 5 个 benchmark 优化问题进行对比实验. 无特殊说明, 所有算法中 w 选择在 [0.9,0.4] 之间随代数线性递减. c_1 和 c_2 均为 1, DDPSO 的扰动参数 $P_v = 0.12$, VCDPSO 算法的最大速度限制值的内部协调参数为 1.2, 本文 MSCMDPSO 算法 $M = 5$, 初始多尺度概率变异算子 p^0 随机设定为 0~1 的范围, 每一个尺度的克隆规模为 20, 所有实验的种群规模为 20, 函数维度设置为 20, 每一位变量的编码长度为 20, 每次运行 2 000 代. 为了排除算法内部随机操作对性能的影响, 对每一个函数独立运行 30 次试验并对其统计结果进行分析.

本文选择 DPSO 算法经常使用的 5 个 benchmark 函数问题进行数值实验, 并根据函数性质分为具有单一极小点 (单模态) 和多个极小点 (多模态) 两大类, 下列公式给出了函数的定义及取值范围和全局最优解.

单模态函数定义如下:

DeJong 函数

$$f_1 = \sum_{i=1}^{n} x_i^2, \tag{8}$$

取值范围为 [−50, 50], 维度为 20, 最小值和位置为 $0(0, \cdots, 0)$.

Rosenbrock 函数

$$f_2 = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2) + (x_i - 1)^2), \tag{9}$$

取值范围为 [−50, 50], 维度为 20, 最小值和位置为 $0(1, \cdots, 1)$.

多模态函数定义如下:

Griewank 函数

$$f_3 = \frac{1}{4000}\sum_{i=1}^{n} (x_i)^2 - \prod_{i=1}^{n} \cos((x_i/\sqrt{i})) + 1, \tag{10}$$

取值范围为 [−50, 50], 维度为 20, 最小值和位置为 $0(0, \cdots, 0)$,

Rastrigrin 函数

$$f_4 = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10), \tag{11}$$

取值范围为 [−50, 50], 维度为 20, 最小值和位置为 $0(0, \cdots, 0)$.

Ackley 函数

$$f_5 = -20\exp\Big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\Big) - \exp\Big[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\Big] + 20 + e, \tag{12}$$

取值范围为 [−50, 50], 维度为 20, 最小值和位置为 $0\,(0, \cdots, 0)$.

函数 f_1 和 f_2 是单模态函数, 其中 Rosenbrock 函数是一个经典的复杂优化问题, 它的全局最优点位于一个平滑、狭长的抛物线形山谷内. 由于函数为优化算法, 只提供了少量信息, 使得算法很难辨别搜索方向, 以至于找到全局最小点的机会微乎其微. 因此, 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[2].

5.2　单模态函数优化性能对比实验

为了验证本文提出的算法针对单模态函数优化问题的有效性, 首先对 DeJong 函数进行优化性能对比实验, 实验结果如图 3 所示. 从图 3 的结果可以看出, 本文算法随着迭代次数的增加迅速向着最优解区域靠近, 体现了大尺度概率变异算子强劲的全局搜索能力, 同时利用小尺度概率变异算子实现了局部精确

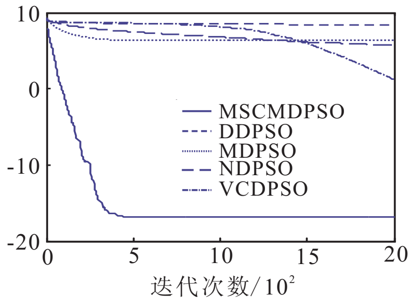

图 3 不同 **DPSO** 算法 **DeJong** 函数优化性能对比

解的搜索, 大大提高了算法在当前最优解附近的开采能力.

对于复杂单模态 Rosenbrock 函数优化问题, 本文算法同样在算法的初期具有很强的全局解定位能力. 由于 Rosenbrock 函数的山谷仅给算法提供了很少量的信息, 使传统 DPSO 进化算法很难在短时间内辨别搜索方向, 极易陷入局部最优解. 而本文算法采用了多尺度概率克隆变异, 能够有效地逃逸出这些局部极值点, 实现最优解区域的准确定位, 使得算法能在短时间内找到正确的搜索方向, 并具有较快的下降速度. 同时在算法后期, 通过小尺度概率克隆变异使算法在最优解附近进行更加详尽的局部搜索, 提高了算法局部搜索性能及所求解的精度.

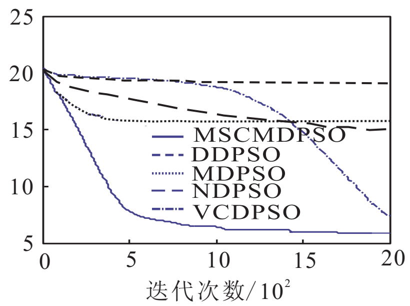

图 4 不同 **DPSO** 算法 **Rosenbrock** 函数优化性能对比

表 1 显示了单模态 benchmark 问题的对比实验结果, 从各个算法的最优值和最差值的实验结果可以看出, 本文提出的算法在所有函数上均取得了较好的实验结果, 具有较好的局部精确解搜索能力. 这是由于本文的 MSCMDPSO 算法通过多尺度概率克隆变异操作能实现勘探能力和开采能力的协调, 增加了算法在最优解附近对局部精确解的搜索能力的同时保持了全局解搜索性能. 实验中还可发现, 由于 DPSO 算法在种群初始化和通过速度确定微粒当前的状态过程中均具有随机性, 使得算法的结果具有一定的随机性. 表 1 的实验结果显示, 对于复杂的 Rosenbrock 函数问题, 很难确定最优解搜索方向, 因此各个算法单次结果存在较大差异. 由于函数的特殊性质以及随机变异的引入, 降低了 DDPSO 算法的搜索性能, 反而陷入了局部极小点. VCDPSO 由于采用了固定的权重系数以及变化的速度极值, 经过一定的迭代也逃出了局部极小解. 而本文提出的 MSCMDPSO 算法能够在算法的初始阶段通过多尺度概率克隆变异逃出局部最优解, 从而定位到全局最优解区域, 找到进化方向, 并通过小尺度概率克隆变异操作进行局部精确解搜索.

5.3 多模态函数优化性能对比实验

对于多模态 Griewank 函数, 本文提出的算法能够在开始阶段寻找到全局最优解区域, 其全局搜索能力明显优于其他算法, 能够有效逃出局部极小点找到全局最小点. 由表 2 可知, MSCMDPSO 算法的中值、平均值和标准方差等各项数据均显著优于其他算法, 显示了新算法的稳定性和健壮性.

多模态 Rastrigrin 函数是一个典型的用来测试算法全局搜索性能的函数, 从图 5 和表 2 可以看出, 本文 MSCMDPSO 算法具有良好的全局搜索性能和较快的搜索速度. 虽然表 2 中本文算法没有在有限次的运行都找到近似全局最优解 0, 但是, 只要给予足够长的迭代次数, MSCMDPSO 算法总能通过多尺度概率克隆变异操作逃出局部极小点, 找到精确的全局最优解. 图 6 显示了 20 维 Ackley 函数的运行结果, 本文的 MSCMDPSO 算法同样能在算法的初期寻找到全局最

表 1 **MSCMDPSO** 和其他 **DPSO** 算法在单模态 **benchmark** 问题上的数值对比

Function	Algorithm	Min	Median	Mean	Deviation	Max
Delong	DPSO	3.118 1e+003	4.529 7e+003	4.503 6e+003	543.711 0	5.384 7e+003
	DDPSO	3.221 0e+003	4.478 8e+003	4.507 8e+003	586.212 7	6.027 4e+003
	MDPSO	281.187 9	583.371 5	555.769 3	143.668 0	825.065 8
	NDPSO	128.504 8	265.434 8	296.190 4	133.168 2	623.357 8
	VCDPSO	0.946 2	3.450 7	3.494 9	1.556 2	7.562 8
	MSCMDPSO	**4.547 5e-008**	**4.547 5e-008**	**4.547 5e-008**	**0**	**4.547 5e-008**
Rosenbrock	DPSO	8.629 2e+007	1.975 1e+008	1.977 7e+008	5.627 3e+007	3.169 6e+008
	DDPSO	6.807 0e+007	2.114 1e+008	2.014 7e+008	4.995 7e+007	2.762 9e+008
	MDPSO	1.705 3e+006	5.657 1e+006	6.502 7e+006	4.923 7e+006	2.330 7e+007
	NDPSO	5.521 4e+005	1.929 6e+006	3.087 2e+006	3.349 4e+006	1.375 9e+007
	VCDPSO	342.196 7	913.018 0	1.637 7e+003	1.543 4e+003	6.143 2e+003
	MSCMDPSO	**17.481 8**	**77.896 7**	**360.132 2**	**773.495 5**	**4.112 0e+003**

表 2　MSCMDPSO 和其他 DPSO 算法在多模态 benchmark 问题上的数值对比

Function	Algorithm	Min	Median	Mean	Deviation	Max
Griewank	DPSO	1.727 2	2.102 8	2.084 6	0.163 2	2.275 3
	DDPSO	1.767 5	2.176 1	2.149 5	0.119 9	2.328 5
	MDPSO	1.062 0	1.144 8	1.147 4	0.052 0	1.321 1
	NDPSO	0.992 3	1.072 3	1.070 3	0.031 2	1.131 2
	VCDPSO	0.084 2	0.229 9	0.233 9	0.097 1	0.453 8
	MSCMDPSO	**4.101 5e-009**	**0.011 3**	**0.019 0**	**0.026 0**	**0.123 5**
Rastrigrin	DPSO	3.201 0e+003	4.610 8e+003	4.571 3e+003	474.255 8	5.227 5e+003
	DDPSO	3.663 5e+003	4.894 1e+003	4.850 5e+003	430.525 4	5.515 4e+003
	MDPSO	506.295 8	698.582 8	743.314 6	173.983 1	1.218 8e+003
	NDPSO	279.232 3	438.166 6	470.654 4	142.222 2	829.938 5
	VCDPSO	50.994 7	72.357 5	74.469 3	13.435 8	101.982 7
	MSCMDPSO	**8.382 5**	**24.643 5**	**25.857 2**	**9.856 3**	**48.965 3**
Ackley	DPSO	8.629 2e+007	1.975 1e+008	1.977 7e+008	5.627 3e+007	3.169 6e+008
	DDPSO	20.080 0	20.563 5	20.515 7	0.164 5	20.713 0
	MDPSO	13.327 0	17.425 0	17.402 6	1.705 5	20.624 2
	NDPSO	10.592 9	14.127 2	14.665 0	2.896 3	20.009 9
	VCDPSO	1.642 5	2.850 3	2.934 7	0.625 9	4.574 5
	MSCMDPSO	**1.908 6e-004**	**1.908 6e-004**	**1.908 6e-004**	**0**	**1.908 6e-004**

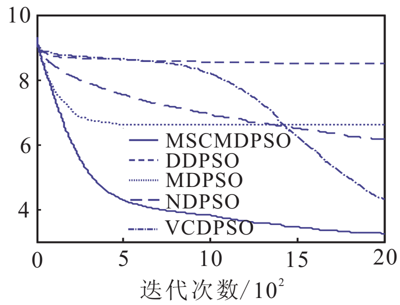

图 5　不同 DPSO 算法 Rastrigrin 函数优化性能对比

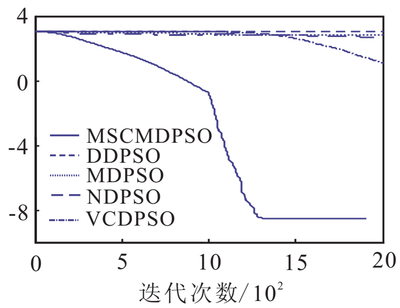

图 6　不同 DPSO 算法 Ackley 函数优化性能对比

优解区域, 然后利用 DPSO 进化和小尺度概率克隆变异操作实现最优解附近的局部精确解搜索.

从表 2 的结果可以看出, 本文提出的算法在各个方面都大大改善了算法全局解寻优能力, 并提高了最优解精度. 这是由于本文算法采用了多尺度概率克隆变异算子, 使得算法能在搜索空间内进行分散式的搜索, 加快全局最优解的搜索速度的同时, 在进化后期有效实现最优解的深度搜索. 从表 2 中最优解和最差解实验结果之间的差异可以看出, 无论是 DPSO 还是其他改进算法都不具备稳定性, 而本文算法相对于其他算法而言稳定性大大提高, 这是因为算法在进化前期能迅速定位到全局最优解区域.

6　结　论

1) 针对传统离散微粒群算法状态转移过程中存在最优解区域局部搜索能力差的不足, 提出一个多尺度概率变异算子, 该算子可以有效实现在最优解附近精确的局部搜索, 同时在算法初期利用大尺度概率克隆变异操作可以使微粒群进行发散式的全局搜索, 从而快速定位到全局最优解区域; 算法后期, 在小尺度概率克隆变异操作进行局部精确解搜索的同时, 利用大尺度概率变异的震荡性质可以帮助算法逃逸出局部极小解, 实现算法勘探及开采能力的有机协调.

2) 采用本文算法及其改进算法, 对 5 个经典 benchmark 函数进行优化, 实验结果表明本文算法在全局搜索性能及最优解精度方面都有显著提高. 需要指出的是, 本文算法的计算复杂度相对其他算法而言略有增加, 但这也是符合智能算法中"没有免费午餐"定理的.

参考文献(References)

[1] Poli R, Kennedy J, Blackwell T. Particle swarm optimization: An overview[J]. Swarm Intell, 2007, 1(1): 33-57.

[2] 陶新民, 徐晶. 改进的多种群协同进化微粒群优化算法[J]. 控制与决策, 2009, 24(9): 1406-1411
(Tao X M, Xu J. Multi-species cooperative particle swarm optimization algorithm[J]. Control and Decision, 2009, 24(9): 1406-1411.)

[3] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]. Proc of the World Multiconference on Systemics, Cybemetics and Informatics. Orlando, 1997: 4104-4109.

[4] Pan Q K, Tasgetiren M F, Liang Y C. A discrete particle

swarm optimization algorithm for the no-wait flow shop scheduling problem[J]. Computers & Operations Research, 2008, 35(9): 2807-2839.

[5] Liu Y, Gu X P. Skeleton-network reconfiguration based on topological characteristics of scale-free networks and discrete particle swarm optimization[J]. IEEE Trans on Power Systems, 2007, 22(3): 1267-1274.

[6] 潘全科, 王凌, 高亮. 离散微粒群优化算法的研究进展[J]. 控制与决策, 2009, 24(10): 1441-1449.
(Pan Q K, Wang L, Gao L. The state-of-art of discrete particle swarm optimization algorithms[J]. Control and Decision, 2009, 24(10): 1441-1449.)

[7] 张国富, 蒋建国, 夏娜. 基于离散粒子群算法求解复杂联盟生成问题[J]. 电子学报, 2007, 35(2): 323-327.
(Zhang G F, Jiang J G, Xia N. Solutions of complicated coalition generation based on discrete particle swarm optimization[J]. Acta Electronica Sinica, 2007, 35(2): 323-327.)

[8] 钟一文, 宁正元, 蔡荣英. 一种改进的离散粒子群优化算法[J]. 小型微型计算机系统, 2006, 27(10): 1893-1896.
(Zhong Y W, Ning Z Y, Cai R Y. An improved dscrete particle swarm optimization algorithm[J]. Mini-micro Systems, 2006, 27(10): 1893-1896.)

[9] 黄艳新, 周春光. 一种求解类覆盖问题的混合算法[J]. 软件学报, 2005, 16(4): 513-522.
(Huang Y X, Zhou C G. A hybrid algorithm on class cover problems[J]. J of Software, 2005, 16(4): 513-522.)

[10] Alberto G V, Rafael P. Introducing dynamic diversity into a discrete particle swarm optimization[J]. Computer and Operation Research, 2009, 36(3): 951-966.

[11] 钟一文, 蔡荣英. 求解二次分配问题的离散粒子群优化算法[J]. 自动化学报, 2006, 27(10): 1893-1896.
(Zhong Y W, Cai R Y. Discrete particle swarm optimization algorithm for QAP[J]. Acta Automatica Sinica, 2006, 27(10): 1893-1896.)

[12] 吕强, 汤贤铭, 俞金寿. 基于信息素机制的离散粒子群算法及其应用[J]. 系统仿真学报, 2008, 20(2): 395-398.
(Lv Q, Tang X M, Yu J S. Discrete particle swarm optimization algorithm and its application based on pheromone mechanism[J]. J of System Simulation, 2008, 20(2): 395-398.)

[13] 余伶俐, 蔡自兴. 改进混合离散粒子群的多种优化策略算法[J]. 中南大学学报, 2009, 40(4): 1047-1053.
(Yu L L, Cai Z X. Multiple optimization strategies for improving hybrid discrete particle swarm[J]. J of Central South University, 2009, 40(4): 1047-1053.)

[14] Khanesar M A, Teshnehlab M, Shoorehdeli M A. A novel binary particle swarm optimization[C]. Proc of the 15th Mediterranean Conf on Control & Automation. Athens-Greece, 2007: 1-6.

[15] 许金友, 李文立, 王建军. 离散粒子群算法的发散性分析及其改进研究[J]. 系统仿真学报, 2009, 21(15): 4676-4681.
(Xu J Y, Li W L, Wang J J. Research on divergence analysis of discrete particle swarm optimization algorithm and its modification[J]. J of System Simulation, 2009, 21(15): 4676-4681.)

(上接第699页)

[7] Ramin Eslami, Hayder Radha. Wavelet-based Contourlet transform and its application to image coding[C]. Proc of Int Conf on Image Processing. Singapore, 2004, 5: 3189-3192.

[8] Do M N. Directional multiresolution image representation[D]. Switzerland: Department of Communication Systems, Swiss Federal Institute of Technology, 2001.

[9] Duncan D Y Po, Minh N Do. Directional multiscale modeling of images using the Contourlet transform[J]. IEEE Trans on Image Processing, 2006, 15(6): 1610-1620.

[10] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, et al. Image quality assessment: From error visibility to structural similarity[J]. IEEE Trans on Image Processing, 2004, 13(4): 600-612.

[11] Maes F, Collignon A, Vandermeulen D, et al. Multimodality image registration by maximization of mutual information[J]. IEEE Trans on Medical Imaging, 1997, 16(2): 187-198.

[12] Piella G. A general framework for multiresolution image fusion: From pixels to regions[J]. Information Fusion, 2003, 4(4): 259-280.

[13] Xydeas C S, Petrovid V. Objective image fusion performance measure [J]. Electronics Letters, 2000, 36(4): 308-309.