文章编号: 1001-0920 (2016) 07-1185-07 DOI: 10.13195/j.kzyjc.2015.0752

动态小生境半径两阶段多模态差分进化算法

张贵军, 陈 铭, 周晓根

(浙江工业大学 信息工程学院，杭州 310023)

摘 要: 针对多模态优化问题, 提出一种动态小生境半径两阶段多模态差分进化算法. 基于构象空间退火思想, 设计一种两阶段退火策略来动态调整小生境半径, 并根据退火过程将整个优化过程分为两个阶段. 在第 1 阶段, 通过差分限制变异策略生成高质量的新个体来维持种群的多样性, 促进多模收敛; 在第 2 阶段, 利用种子邻近变异策略对已探测到的生境高度搜索, 加快算法的收敛速度. 实验结果表明, 所提出算法能够有效实现从全局探测到局部增强的自适应平滑过渡, 是一种有效的多模态优化算法.

关键词: 差分进化；多模态优化；小生境；两阶段优化；构象空间退火

中图分类号: TP301 **文献标志码:** A

Two-stage differential evolution algorithm using dynamic niche radius for multimodal optimization

ZHANG Gui-jun, *CHEN Ming*, *ZHOU Xiao-gen*

(College of Information Engineering，Zhejiang University of Technology，Hangzhou 310023，China．Correspondent: ZHANG Gui-jun，E-mail：zgj@zjut.edu.cn)

Abstract: A two-stage differential evolution algorithm using dynamic niche radius is proposed for multimodal optimization, in which a two-stage annealing schedule based on the idea of conformational space annealing is designed to adjust the niche radius dynamically. Meanwhile, the optimization process is divided into two stages according to the annealing process. Thus, at the first stage, a differential vector limited mutation is used to generate the high-quality individuals to keep the population diversity, thereby facilitating multiple convergence. At the second stage, to enhance the convergence speed, a seed neighborhood mutation strategy is used to exploit the niche highly. Experiment results show that, the proposed algorithm can navigate from global exploration to local exploitation adaptively, which is an effective multimodal optimization algorithm.

Keywords: differential evolution；multimodal optimization；niching；two-stage optimization；conformational space annealing

0 引 言

在实际应用中许多优化问题都是多模态优化问题, 即问题存在一个或多个全局最优解以及若干个局部最优解. 在蛋白质结构预测、机械设计、电磁设计以及电力系统规划等领域, 除了需要得到一个全局最优解以外, 还需要得到其他全局最优解和一些高质量的局部最优解. 进化算法通过模拟达尔文的生物进化过程和机制来解决优化问题, 典型的进化算法包括差分进化算法(DE)[1]、遗传算法(GA)[2]、进化策略(ES)[3] 和粒子群算法(PSO)[4], 这些算法虽然可以有效地求得问题的某一个全局最优解, 但是由于选择压力[5] 的存在, 使其无法得到其他全局最优解和一些高质量的局部最优解.

为了解决多模态优化问题, 国内外学者借鉴生物学中小生境的概念, 并结合DE、GA 和 PSO 等进化算法, 相继提出了各种小生境多模态优化算法[6-8]. Thomsen[9]提出了排挤差分进化算法 (CDE) 和适应度共享差分进化算法 (SharingDE). 在 CDE 算法的选择过程中, 将新产生的个体与其欧氏距离最近的种群个体比较; 在 SharingDE 算法中, 根据个体的相似度将

收稿日期: 2015-06-11；**修回日期:** 2015-08-11.

基金项目: 国家自然科学基金项目(61075062, 61573317)；浙江省自然科学基金项目(LY13F030008)；浙江省科技厅公益项目(2014C33088)；浙江省公益技术研究社会发展重点项目(2015C33001)；浙江省重中之重学科开放基金项目(20151108, 20151015).

作者简介: 张贵军(1974−), 男, 教授, 博士生导师, 从事智能信息处理、优化理论及算法设计、生物信息学等研究；陈铭(1990−), 男, 硕士生, 从事智能优化的研究.

种群划分为多个子种群, 然后在同一生境中共享各个体的适应度信息. Li[10]提出一种物种形成差分进化算法(SDE), 该算法首先将初始种群划分为多个物种, 然后在各物种内执行DE 算法进行搜索. Qu 等[11]提出一种邻近变异策略, 并将此策略应用到CDE、Sharing DE 和SDE 算法中; 在邻近变异策略中, 差分个体的选取被限制在一系列欧氏距离相近的种群个体中, 因此, 算法可以高度探测各搜索子区域. Zhang 等[12]提出一种基于广义凸下界估计的多模态差分进化算法(ACDE), 该算法通过建立目标函数的广义凸下界估计模型获取上下界偏差值来评价各生境的拥挤程度, 从而保证形成协同进化的种群.

上述小生境算法取得了一定的效果. 然而, 在小生境方法中, 由于目标函数曲面的复杂性, 小生境半径的选择直接影响着算法的性能. 为了确定合适的生境半径值, Deb 等[13]提出了一种参数估计方法. 然而, 由于目标问题的适应度曲面很难预知, 而且即使知道目标问题的适应度分布情况, 对于绝大数实际问题, 其极值解分布也极不均匀, 某些很小的区域可能散布了大量的极值解, 而其他一些大范围区域则可能只包含很少一部分极值解, 因此, 使用一个固定的生境半径并不合适[14]. 在此情形下, 若要得到问题的所有极值解, 则必须把生境半径确定的非常小, 使得即使相距很近的两个生境也可以准确地探测到. 然而, 当生境半径过小时, 往往会导致形成过多的生境, 反而不能达到应有的效果; 反之, 如果生境半径过大, 则无法探测到比设置的半径小的生境. 因此, 如何确定一个合适的生境半径值是一项富有挑战性的工作.

针对上述问题, 本文基于构象空间退火思想[15], 提出一种动态小生境半径两阶段多模态差分进化算法(DNTDE), 通过设计一种两阶段退火模型来动态调整小生境半径. 第1阶段, 生境半径值从一个较大的值开始随着设定的退火曲线逐渐减小, 从而保证探测到尽可能多的生境, 同时通过一种差分限制变异策略生成高质量的新个体, 维持种群的多样性, 促进多模收敛; 第2阶段, 当生境半径减小到设定的阈值以后保持不变, 算法开始对已探测到的生境进行局部增强, 并利用一种种子邻近变异策略对已探测到的生境高度搜索, 从而防止模态丢失, 同时提高算法的局部搜索能力, 加快算法的收敛速度, 有效地实现算法从全局搜索到局部增强的动态自适应平滑过渡.

1 动态小生境半径两阶段多模态DE 算法

1.1 两阶段退火策略

一般来讲, 多模态优化算法需要重点解决两个问题: 1) 同峰极值点判断问题, 即如何判别新发现的极值点是否与已求出的极值点属于同一峰; 2) 不同峰极值点保持问题, 即如何将处于不同峰的极值点能够在后续的进化过程中稳定地保存下来. 为了较好地平衡这对矛盾, 本文通过设计一种两阶段退火模型动态调整生境半径, 并根据生境半径的动态变化将整个优化过程分为两个阶段. 在起始阶段, 通过设计好的退火曲线动态调整生境半径, 随着生境半径的减小, 对于小于当前半径值的种群, 算法起全局探测作用, 而对于大于当前半径值的种群, 则起到局部增强作用; 在后续阶段, 当生境半径减小到设定的阈值时则保持不变, 由于生境半径减小至某个相对较小的值, 在确保已探测到的模态不丢失的前提下, 还能够对已探测到的模态起到局部增强的效果. 具体退火策略如下:

$$s = \left(\frac{d_{\rm cut}}{d_{\rm initial}}\right)^{\frac{1}{e_{\rm cut}}}, \tag{1}$$

$$r = d_{\rm initial} s^{\rm FEs}. \tag{2}$$

其中: $d_{\rm cut}$ 为算法从全局探测到局部增强时的生境半径阈值; FEs 为函数评价次数; $e_{\rm cut}$ 为算法从全局探测到局部增强时函数评价次数阈值, 一般设置为总函数评价次数的1/3; $d_{\rm initial}$ 为生境半径初始值, 即

$$d_{\rm initial} = \frac{\sum\limits_{i=1}^{N_P}\sum\limits_{k=i+1}^{N_P}||x_i - x_k||}{\frac{N_P(N_P-1)}{2}}. \tag{3}$$

则两阶段退火策略为

$$r_d = \begin{cases} r, \ r > d_{\rm cut}; \\ d_{\rm cut}, \ \text{otherwise}. \end{cases} \tag{4}$$

即在第1阶段, 生境半径 r_d 等于式(2)的动态半径 r; 在第2阶段, 当生境半径减小到设定的阈值 $d_{\rm cut}$ 时, 保持 $d_{\rm cut}$ 不变.

1.2 两阶段变异策略

1.2.1 差分限制变异策略

在基本DE 算法中[1,16], 通过扰动整个种群中的所有个体来产生新个体, 也就是说, 算法允许种群中的任意两个个体(即使两者相距很近)作为差分向量来产生新个体. 对于求解单个全局最优解问题, 传统的变异策略非常有效, 可以保证所有种群个体最后收敛到一个全局最优解. 然而, 对于多模态优化问题, 为了找到尽可能多的最优解, 算法需要维持种群的多样性, 保证多模收敛, 从而找出尽可能多的模态, 因此, 传统的变异策略并不适合于多模态优化问题. 为了获取高质量的个体, 维持种群的多样性, 促进多模收敛, 本文在算法第1阶段(即全局探测生境阶段)利用一种差分限制变异策略产生新个体. 在此策略中, 根据欧氏距离来衡量差分个体的相似性, 从而根据动态生境半径限制差分个体的选择, 避免选择相似性较高的个体作为差分个体, 具体差分限制变异策略如下:

$$v_i = x_{r_1} + F(x_{r_2} - x_{r_3}),\ ||x_{r_2} - x_{r_3}|| > r_d. \tag{5}$$

其中: $r_1 \neq r_2 \neq r_3$, $||x_{r_2} - x_{r_3}||$ 为差分个体 x_{r_2} 和 x_{r_3} 之间的欧氏距离. 即根据差分个体之间的欧氏距离与动态生境半径的比较, 选取欧氏距离大于生境半径的个体作为差分个体, 避免了因差分向量过小而导致生成的新个体与其他种群个体的相似性较高, 从而可提高新个体的质量, 维持种群的多样性, 保证算法在第 1 阶段找出尽可能多的模态.

1.2.2　种子邻近变异策略

在 DNTDE 算法的第 2 阶段, 小生境半径减小到设定的阈值 d_{cut} 后保持不变, 此时, 算法开始对已探测的生境进行局部增强. 为了防止已探测到的模态丢失, 并使得各种群个体向对应的峰值快速收敛, 在第 2 阶段, 引入种子邻近变异策略对已探测到的生境进行高度搜索, 即

$$v_i = x_{\text{seed}} + K(x_{\text{seed}} - x_{\text{neigh}}). \tag{6}$$

其中: x_{seed} 为种子个体, x_{neigh} 为种子个体的邻近个体 (根据欧氏距离选取), $K \in [-0.5, 1.5]$ 为增益常数. 当 $K > 1$ 时, 算法在远离种子的区域进行探测, 当 $K < 0$ 时, 算法则在靠近种子的区域进行搜索.

如图 1 所示的模态示意图, 假设 A 为种子个体, B 为 A 的邻近个体, C 为式 (6) 在 $K = 1.5$ 时生成的新个体, D 为在 $K = -0.5$ 时生成的新个体, 可以看出, $K = -0.5$ 时生成的个体 D 更接近于最优解; 当根据邻近个体 E 生成个体时, F 为式 (6) 在 $K = 1.5$ 时生成的新个体, G 为在 $K = -0.5$ 时生成的新个体, 此时, $K = 1.5$ 时生成的个体 F 最优, 更接近最优解. 因此, 在 DNTDE 算法第 2 阶段, 选择种子个体的 n 个邻近个体, 并根据式 (6) 分别生成 $K = 1.5$ 和 $K = -0.5$ 时的新个体, 然后对生成的 $2n$ 个新个体进行更新操作, 从而达到对已探测到的生境高度搜索的效果, 加快了算法的收敛速度.

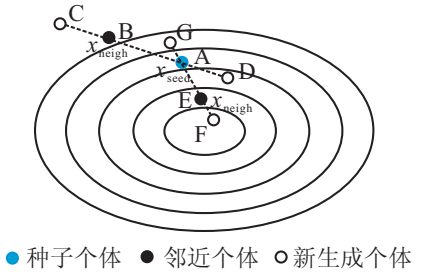

图 1　种子邻近变异策略示意图

1.3　种子选取

种子的选取是一个重要的环节, 既要确保新产生的种子与之前的种子距离足够远, 以保证所有个体不会收敛至同一个最优解, 又要确保产生的新种子有利于维持种群的多样性. 考虑到实际应用中目标问题的极值解分布往往极不均匀, 因而提出一种新的种子选取策略. 首先, 将整个种群设置为自由个体 (即未被标记为种子), 根据各个体的目标函数值降序排序, 以种群中最优个体作为初始种子, 然后计算出各自由个体与种子个体的平均距离 d_{ave}, 即

$$d_{\text{ave}} = \frac{\sum_{i=1}^{N_P} ||x_i - x_{\text{seed}}^{\text{pre}}||}{N_F}. \tag{7}$$

其中: $x_{\text{seed}}^{\text{pre}}$ 为当前种子, N_F 为自由个体数目, 进而根据平均距离选取新种子.

$$x_{\text{seed}}^{\text{new}} = \begin{cases} x_i, & ||x_i - x_{\text{seed}}^{\text{pre}}|| > d_{\text{ave}}; \\ x_{\text{seed}}^{\text{pre}}, & \text{otherwise}. \end{cases} \tag{8}$$

如果某种群个体 x_i 为自由个体, 且与当前种子 $x_{\text{seed}}^{\text{pre}}$ 之间的距离大于 d_{ave}, 则将其选为新种子, 否则保持当前种子不变.

1.4　种群更新

传统 DE 算法的选择策略对求解单个全局最优解的优化问题极其有效, 但是对于多模态优化问题, 为了能够有效地保存已探测到的模态, 本文根据动态小生境半径来更新种群.

如图 2 所示, 在更新过程中, 可能会出现两种情况: 1) 假设 B 为新个体, C 为与其最近的种群个体, 因为 B 与 C 的距离 d_{bc} 小于生境半径 r_d, 故比较 B 与 C 个体的目标函数值, 因为 B 个体优于 C 个体, 故用 B 个体替换 C 个体; 2) 假设 E 为新个体, D 为与其最近的种群个体, 因为 E 与 D 之间的距离 d_{ed} 大于生境半径 r_d, 故将 E 个体与整个种群中最差的个体 F 进行比较; 因为 E 个体优于最差个体 F, 故用 E 个体替换 F 个体. 这样, 整个种群的规模保持不变. 需要注意的是, 生境半径 r_d 是随着退火曲线动态减小的, 直到达到阈值 d_{cut} 为止保持不变, 且会在每代进化中重新计算.

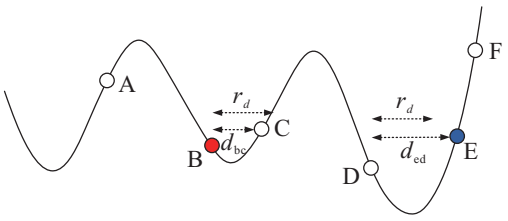

图 2　种群更新示意图

1.5　算法描述

整个算法整体流程如下.

Step 1: 初始化种群, 并对各个体进行目标函数评价.

Step 2: 判断是否满足终止条件.

Step 3: 将种群中最优个体设置为初始种子, 并将其他个体设置为自由个体.

Step 4: 判断种群中个体是否全为种子, 如果是则

返回 Step 3, 否则继续 Step 5.

Step 5: 若生境半径 $r_d > d_{\rm cut}$, 则根据式 (5) 的差分限制变异策略生成新个体; 若 $r_d = d_{\rm cut}$, 则根据式 (6) 的种子邻近变异策略生成新个体.

Step 6: 根据种群更新策略更新种群.

Step 7: 根据种子选取策略选取新种子.

Step 8: 如果种群中所有个体都为种子, 则一轮结束, 并返回 Step 2, 否则返回 Step 4.

根据上述步骤进行时间复杂度分析. 在 Step 3 中找出最优个体的时间复杂度为 $O(N_P)$; 在 Step 4 中判断种群个体是否全为种子的时间复杂度为 $O(N_P)$; 在 Step 5 中判断生境半径 r_d 是否大于 $d_{\rm cut}$ 的时间复杂度为 $O(1)$; 在 Step 6 更新策略中找出与新个体最近的个体的时间复杂度为 $O((N+1)N_P)$, N 为问题维数, 计算新个体与最近个体的距离的时间复杂度为 $O(N)$; 在 Step 7 的种子选取策略中计算平均距离的时间复杂度为 $O(NN_P)$, 找出新种子的时间复杂度为 $O(N_P)$. 忽略常量、低次幂和最高次幂的系数, 最终时间复杂度为 $O(NN_P)$. 可以看出, DNTDE 算法没有带来较高的时间复杂度.

2 数值实验

2.1 测试函数及参数设置

为了验证本文算法的有效性, 选用多个函数进行测试. 表 1 给出了各函数的基本参数, 各函数的具体数学表达式见文献 [17].

表 1 多模态测试函数

函 数 名	维数	搜索范围	优化数
f_1: Central Two-Peak Trap	1	$[0, 20]$	2
f_2: Five-Uneven-Peak Trap	1	$[0, 30]$	4
f_3: Branin RCOS	2	$[-5, 10], [0, 15]$	3
f_4: Himmelblau's function	2	$[-6, 6]$	4
f_5: Inverted Shubert function	2	$[-10, 10]$	18
f_6: Problem Peaks5	10	$[-100, 100]$	4
f_7: Griewank's function	10	$[-100, 100]$	1
f_8: Ursem F1	2	$[-2.5, 3], [-2, 2]$	2
f_9: Ursem F3	2	$[-2.5, 3], [-2, 2]$	5
f_{10}:Ursem F4	2	$[-2.5, 3], [-2, 2]$	5

实验中, 选取近年来提出的主流多模态优化算法进行比较: 基于适应度欧氏距离比的差分进化算法 (FERDE)[18], 排挤差分进化算法 (CDE)[9], 拓扑物种保存多模态优化算法 (TSC2)[14]和 4 种多模态粒子群算法 (r2pso, r3pso, r2pso-lhc 和 r3pso-lhc)[19]. DNTDE 算法参数设置为: $CR = 0.1, F = 0.5, N_P = 100$, 种子邻近变异策略中邻近个体的数目 $n = 3$, 生境阈值 $d_{\rm cut} = 2.0$; 其他比较算法的参数设置均采用原文献中的设置. 为了公平比较, 所有算法的终止条件均为 FEs = 30 000, 各算法对各测试函数均独立运行 30 次. 实验环境为: Intel(R) Core i5-2410M CPU@2.30 GHz with 8GB RAM, Windows 7, 算法采用 Matlab 8.2 和 Visual Studio 2012 C++ 编写.

2.2 评价指标

采用以下性能指标进行评价.

1) 峰比值 (Pr): 探测到的峰值点与峰值点总数的比值, 其中规定所探测到的峰值与实际峰值的误差 $\delta \leqslant 0.1$ 时为成功.

2) 峰值精度 (Pa): 已探测到的峰值点与最近的实际峰值点的适应度值绝对值之和, 即

$$\mathrm{Pa} = \sum_{i=1}^{\#\mathrm{peaks}} |f(\mathrm{peak}_i) - f(x)|. \tag{9}$$

3) 距离精度 (Da): 如果某些峰值点的值相同, 且各峰值点相距很近, 则即使只探测到其中一个或几个峰值点, 峰值精度也会很高. 为了防止潜在的误差, 本文进一步引入距离精度进行评价, 其计算公式与峰值精度相似, 只需将式 (9) 中的目标函数的绝对值替换为欧氏距离即可.

2.3 实验结果及分析

表 2 列出了各测试函数 30 次独立运行的峰比值. 从表 2 中结果可以看出, 除了函数 f_5 和 f_6 外, DNTDE 算法可以找到其他函数所有的最优解. 对于函数 f_1, 只有 r2pso、r2pso-lhc 和 r3pso-lhc 算法无法找到其全局最优解; 对于函数 f_2, 只有 r3pso、r2pso-lhc 和 r3pso-lhc 算法无法成功求解; 对于函数 f_3, 仅有 DNTDE、FERDE 和 r2pso 算法能够对其成功求解; 对于函数 f_4, 除了 TSC2 和 r3pso-lhc 算法略逊外, 其余算法均能成功求解; 对于函数 f_5, 只有 CDE 算法能够找到所有的全局最优解, DNTDE 可以找到 98% 的全局最优解; 对于高维函数 f_6, 因其函数曲面的复杂性, 所有算法在函数评价次数 30 000 次内均未找到其全局最优解; 对于高维函数 f_7, 只有 DNTDE 算法的峰值比最高为 1, CDE 算法为 0.27, 其余算法均为 0; 对于函数 f_8, 只有 TSC2、r2pso 和 r3pso 算法未能成功求解; 对于函数 f_9, 仅有 DNTDE、FERDE、CDE 和 TSC2 算法能够找到其所有解; 对于函数 f_{10}, 仅有 DNTDE、CDE 和 FERDE 算法能够成功求解. 综合来看, DNTDE 算法的峰值比最优, 对于绝大多数函数能够找到所有的全局最优解.

表 3 给出了各测试函数 30 次独立运行的平均峰值精度和最优峰值精度, 其中最优结果通过加粗标出. 由表 3 中数据可以明显看出, 对于函数 f_1、f_2、$f_5 \sim f_9$, DNTDE 算法在峰值精度方面优于其他 7 种算法, 尤其对于函数 f_7, DNTDE 算法优于其他 6 种算法

表 2　各测试函数的峰比值 (Pr), 平均 (最优)

函数	维数	DNTDE	FERDE	CDE	TSC2	r2pso	r3pso	r2pso-lhc	r3pso-lhc
f_1	1	1(1)	1(1)	1(1)	1(1)	0.97(1)	1(1)	0.93(1)	0.77(1)
f_2	1	1(1)	1(1)	1(1)	1(1)	1(1)	0.96(1)	0.96(1)	0.96(1)
f_3	2	1(1)	1(1)	0.94(1)	0.60(1)	1(1)	0.93(1)	0.93(1)	0.98(1)
f_4	2	1(1)	1(1)	1(1)	0.95(1)	1(1)	1(1)	1(1)	0.99(1)
f_5	2	0.98(1)	0.99(1)	1(1)	0.13(0.61)	0.85(0.94)	0.84(1)	0.85(1)	0.84(1)
f_6	10	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
f_7	10	1(1)	0(0)	0.27(1)	0(0)	0(0)	0(0)	0(0)	0(0)
f_8	2	1(1)	1(1)	1(1)	0.76(1)	0.96(1)	0.60(1)	1(1)	1(1)
f_9	2	1(1)	1(1)	1(1)	1(1)	0.88(1)	0.53(0.8)	0.92(1)	0.75(1)
f_{10}	2	1(1)	1(1)	1(1)	0.44(1)	0.36(0.4)	0.30(0.4)	0.40(0.6)	0.4(0.6)

表 3　各测试函数的峰值精度 (Pa), 平均 (最优)

函数	维数	DNTDE	FERDE	CDE	TSC2	r2pso	r3pso	r2pso-lhc	r3pso-lhc
f_1	1	**2.15e-10**	6.64e-09	3.43e-08	8.29e-06	5.33e-05	7.25e-06	8.91e-05	2.64e-04
		(**4.87e-10**)	(1.65e-09)	(6.71e-09)	(3.95e-08)	(1.09e-07)	(5.26e-7)	(4.79e-7)	(4.43e-07)
f_2	1	**2.28e-06**	1.40e-05	7.37e-05	4.83e-05	2.17e-04	3.87e-03	2.81e-03	5.25e-03
		(**4.22e-07**)	(8.03e-07)	(5.40e-06)	(1.90e-05)	(1.01e-04)	(2.65e-04)	(3.74e-04)	(3.61e-04)
f_3	2	4.91e-04	**5.41e-05**	2.93e-03	1.25e-02	4.60e-03	3.81e-04	2.18e-04	2.24e-04
		(8.63e-07)	(1.87e-05)	(2.40e-04)	(7.74e-07)	(5.54e-07)	(8.21e-08)	(**8.43e-10**)	(4.01e-08)
f_4	2	4.37e-04	**5.00e-06**	6.21e-04	7.24e-03	2.03e-04	2.32e-05	6.24e-06	1.88e-05
		(1.04e-04)	(1.14e-06)	(1.23e-04)	(2.22e-05)	(2.68e-09)	(6.54e-08)	(5.48e-09)	(**1.49e-09**)
f_5	2	**9.06e-05**	9.80e-02	1.21e-04	1.21e+01	1.11e+00	1.75e+00	9.07e-01	5.98e-01
		(**1.44e-05**)	(1.29e-02)	(1.75e-05)	(4.56e+00)	(3.84e-02)	(1.53e-03)	(2.65e-03)	(5.24e-03)
f_6	10	**6.53e+01**	8.05e+01	1.26e+02	2.45e+02	4.80e+03	4.78e+03	4.53e+03	3.95e+03
		(**3.62e+01**)	(4.81e+01)	(6.24e+01)	(1.81e+02)	(6.26e+02)	(5.98e+02)	(5.85e+02)	(4.26e+02)
f_7	10	**6.39e-04**	1.85e-01	1.01e+01	82.70e+00	4.48e+03	3.39e+03	4.26e+03	3.66e+03
		(**6.43e-05**)	(1.03e-01)	(1.70e-03)	(2.39e+01)	(2.62e+02)	(1.77e+02)	(5.39e+02)	(2.68e+02)
f_8	2	**5.10e-08**	1.88e-03	5.42e-08	1.33e-02	5.43e-03	1.81e-02	2.90e-07	4.53e-07
		(**5.50e-09**)	(3.05e-04)	(6.57e-09)	(1.53e-03)	(2.19e-07)	(2.01e-08)	(9.59e-09)	(6.82e-09)
f_9	2	**1.43e-05**	3.70e-05	1.79e-05	8.04e-05	6.85e-02	4.41e-01	1.17e-01	6.46e-01
		(**2.79e-06**)	(9.00e-06)	(3.43e-06)	(5.80e-06)	(2.23e-03)	(8.72e-03)	(1.80e-03)	(2.77e-03)
f_{10}	2	2.96e-05	**9.01e-06**	1.07e-05	2.36e-01	1.75e+00	2.84e+00	1.48e-01	1.75e+00
		(5.25e-06)	(**1.29e-06**)	(1.85e-06)	(1.26e-02)	(4.32e-01)	(5.37e-02)	(1.25e-02)	(2.87e-02)

表 4　各测试函数的距离精度 (Da), 平均 (最优)

函数	维数	DNTDE	FERDE	CDE	TSC2	r2pso	r3pso	r2pso-lhc	r3pso-lhc
f_1	1	**8.36e-12**	1.74e-10	6.26e-08	3.44e-07	1.43e-04	4.35e-05	7.07e-04	7.13e-03
		(**5.53e-12**)	(5.17e-11)	(3.19e-10)	(5.82e-09)	(4.66e-07)	(3.87e-07)	(2.38e-06)	(2.55e-06)
f_2	1	**5.21e-11**	2.75e-07	2.67e-07	3.77e-06	1.95e-04	3.88e-03	4.85e-03	5.22e-03
		(**3.28e-12**)	(2.51e-08)	(4.89e-08)	(5.06e-07)	(5.43e-05)	(6.12e-05)	(2.62e-04)	(9.11e-04)
f_3	2	1.62e-02	**1.02e-02**	6.41e-02	8.58e-02	4.57e-02	9.92e-02	7.10e-02	3.72e-02
		(1.58e-03)	(6.31e-03)	(2.35e-02)	(1.21e-03)	(1.13e-03)	(3.72e-04)	(3.84e-04)	(**2.89e-04**)
f_4	2	3.45e-02	**6.46e-04**	4.06e-02	8.02e-02	7.32e-03	1.94e-02	1.55e-02	1.92e-02
		(1.78e-02)	(2.95e-04)	(2.01e-02)	(7.80e-03)	(**9.75e-05**)	(5.69e-04)	(1.25e-04)	(1.45e-04)
f_5	2	**1.58e-02**	2.54e-02	3.54e-02	1.17e+01	1.69e+00	3.59e+00	2.48e+00	3.24e+00
		(3.21e-02)	(1.16e-02)	(**1.59e-04**)	(6.55e+00)	(3.72e-01)	(2.84e-01)	(1.87e-01)	(2.59e-01)
f_6	10	**3.07e+01**	3.52e+01	4.23e+01	4.52e+01	1.00e+02	8.54e+01	7.23e+01	6.88e+01
		(**2.34e+01**)	(2.94e+01)	(2.85e+01)	(3.29e+01)	(5.23e+01)	(4.96e+01)	(4.57e+01)	(3.51e+01)
f_7	10	**2.33e-02**	1.26e+01	1.81e+00	8.83e+00	6.28e+01	5.44e+01	6.02e+01	5.68e+01
		(**8.02e-03**)	(6.23e+00)	(4.10e-02)	(4.89e+01)	(1.62e+01)	(1.33e+01)	(2.33e+01)	(1.63e+01)
f_8	2	**2.77e-04**	4.78e-03	2.81e-04	1.06e-01	5.38e-02	3.75e-01	2.67e-03	1.79e-03
		(**1.02e-04**)	(9.61e-04)	(1.08e-04)	(3.92e-02)	(5.47e-04)	(1.88e-04)	(1.29e-04)	(1.16e-04)
f_9	2	**6.71e-03**	6.98e-03	7.51e-03	1.48e-02	2.95e-01	1.66e+00	3.18e-01	1.14e+00
		(**3.40e-03**)	(3.76e-03)	(3.49e-03)	(4.35e-03)	(9.60e-02)	(5.91e-01)	(8.62e-02)	(1.01e-01)
f_{10}	2	9.62e-03	**4.28e-03**	5.74e-03	7.64e-01	2.07e+00	4.22e+00	2.52e+00	3.49e+00
		(3.48e-03)	(**1.40e-03**)	(2.79e-03)	(1.84e-01)	(1.01e+00)	(1.70e+00)	(9.05e-01)	(1.20e+00)

的最优算法(FERDE)3个数量级; 对于函数f_3, 在平均峰值精度方面, FERDE算法获得了最优结果, 而在最优峰值精度方面, r2pso-lhc算法获得了最优结果; 对于函数f_4, FERDE和r3pso-lhc分别在平均峰值精度和最优峰值精度方面获得了最优结果; 对于函数f_{10}, FERD算法在平均和最优峰值精度方面优于其他7种算法, CDE算法次之, FERDE略逊于CDE算法. 综合所有测试函数的结果来看, 无论对于平均峰值精度还是最优峰值精度, DNTDE算法对于绝大多数测试函数表现出了明显优势.

表4为各测试函数30次独立运行的平均距离精度和最优距离精度, 其中最优结果通过加粗标出. 由表4易知, 对于函数f_1、f_2、$f_6 \sim f_9$, DNTDE算法表现出了明显优势, 尤其对于函数f_2, DNTDE算法优于其他所比算法中最优算法(FERDE)4个数量级; 对于函数f_3, FERDE和r3pso-lhc算法分别在平均距离精度和最优距离精度方面优于其他算法, DNTEDE略逊; 对于函数f_4, FERDE和r2pso算法分别在平均距离精度和最优距离精度方面优于其他算法; 对于函数f_5, DNTDE算法在平均距离精度方面优于其他算法, 而CDE算法则在最优距离精度方面优于其他算法; 对于函数f_{10}, FERDE算法表现出了优势, DNTDE算法略逊. 综合来看, 在距离精度方面, DNTDE算法对于绝大多数函数优于其他算法.

为了进一步验证DNTDE算法在峰值精度和距离精度方面的整体优势, 采用“Friedman test”[20]对各测试函数30次独立运行的结果进行参数检验, 其中“rankings”最高的算法为最优算法. 表5给出了检验结果, 其中最优结果通过加粗标出, 可以看出, DNTDE算法无论在峰值距离精度方面, 还是在距离精度方面, 均获得了最高的“rankings”, 峰值精度和距离精度的“rankings”分别为6.9和7.2, FERDE算法的峰值精度和距离精度的“rankings’分别为6.80和6.70, 仅次于DNTDE算法.

表 5 Friedman检验结果

算法	rankings	
	峰值精度	距离精度
DNTDE	**6.90**	**7.20**
FERDE	6.80	6.70
CDE	5.60	5.80
TSC2	3.40	3.70
r2pso	2.75	3.70
r3pso	2.90	2.30
r2pso-lhc	4.20	3.30
r3pso-lhc	3.45	3.30

图3给出了DNTDE算法对函数f_{10}求解时的收敛过程, 根据函数评价次数(FEs)将整个收敛过程划分为4个过程, 其中五角星代表种群个体. 由图3可以看出, DNTDE算法收敛速度很快. 在算法第1阶段, 小生境半径随着退火曲线而动态变化, 此时算法处于全局探测模态阶段. 由图3(b)可以看出, 当函数评价次数达到10 000时, 算法已探测到了所有模态(1个全局最优解和4个位于可行域边界上的次最优解); 当生境半径达到阈值以后保持不变, 进入第2阶段, 算法开始进行局部增强. 由图3(d)可以看出, 当函数评价次数达到30 000时, 所有的个体已经几乎接近各自的最优解.

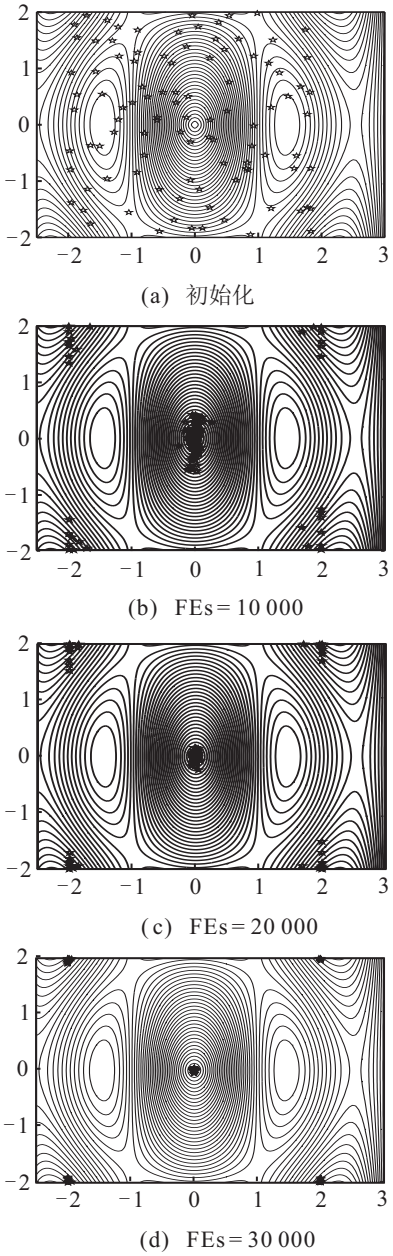

图 3 DNTDE算法对函数f_{10}求解的收敛过程

3 结 论

本文提出了一种动态小生境半径两阶段多模态差分进化算法. 通过设计一种两阶段退火策略来动态调整小生境半径, 并根据小生境半径的变化将算法过程划分为两个阶段: 在第1阶段, 利用差分限制变异策略生成新个体, 从而维持种群多样性, 找出尽可能

多的模态; 在第2阶段, 则利用种子邻近变异策略生成新个体, 从而对种子附近区域高度搜索, 加快算法的收敛速度. 实验结果表明, 所提出算法对于绝大多数测试问题优于其他所比算法. 下一步的工作将集中研究所提出算法中参数(如生境半径阈值)的自适应选择.

参考文献(References)

[1] Storn R, Price K V. Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[2] Young C T, Zheng Y, Yeh C W, et al. Information-guided genetic algorithm approach to the solution of MINLP problems[J]. Industrial & Engineering Chemistry Research, 2007, 46(5): 1527-1537.

[3] Rechenberg I. Evolutions strategie: Optimierung technischer systeme nach prinzipien der biologischen evolution[M]. Stuttgart: Fromman-Holzboog, 1973.

[4] Kennedy J. Particle swarm optimization[M]. New York: Springer, 2010: 760-766.

[5] Rogers A, Prugel-Bennett A. Generic drift in genetic algorithm selection schemes[J]. IEEE Trans on Evolutionary Computation, 1999, 3(4): 298-303.

[6] Basak A, Das S, Tan K C. Multimodal optimization using a biobjective differential evolution algorithm enhanced with mean distance-based selection[J]. IEEE Trans on Evolutionary Computation, 2013, 17(5): 666-685.

[7] Deb K, Saha A. Multimodal optimization using a bi-objective evolutionary algorithm[J]. Evolutionary Computation, 2012, 20(1): 27-62.

[8] Das S, Maity S, Qu B Y, et al. Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art[J]. Swarm and Evolutionary Computation, 2011, 1(2): 71-88.

[9] Thomsen R. Multimodal optimization using crowding-based differential evolution[C]. IEEE Congr Evol Comput. New Jersey: IEEE, 2004: 1382-1389.

[10] Li X. Efficient differential evolution using speciation for multimodal function optimization[C]. The 7th Annual Conf on Genetic and Evolutionary Computation. New York: ACM Press, 2005: 873-880.

[11] Qu B Y, Suganthan P N, Liang J J. Differential evolution with neighborhood mutation for multimodal optimization[J]. IEEE Trans on Evolutionary Computation, 2012, 16(5): 601-614.

[12] 张贵军, 何洋军, 郭海锋, 等. 基于广义凸下界估计的多模态差分进化算法[J]. 软件学报, 2013, 24(6): 1177-1195.
(Zhang G J, He Y J, Guo H F, et al. A differential evolution algorithm for multimodal optimization based on abstract convex underestimation[J]. J of Software, 2013, 24(6): 1177-1195.)

[13] Deb K, Goldberg D E. An investigation of niche and species formation in genetic function optimization[C]. Int Conf on Genetic Algorithms. San Francisco: Morgan Kaufmann Publishers, 1989: 42-50.

[14] Stoean C, Preuss M, Stoean R, et al. Multimodal optimization by means of a topological species conservation algorithm[J]. IEEE Trans on Evolutionary Computation, 2010, 14(6): 842-864.

[15] Lee J, Scheraga H A, Rackovsky S. New optimization method for conformational energy calculations on polypeptides: Conformational space annealing[J]. J of Computational Chemistry, 1997, 18(9): 1222-1232.

[16] 张春美, 陈杰, 辛斌. 参数适应性分布式差分进化算法[J]. 控制与决策, 2014, 29(4): 701-706.
(Zhang C M, Chen J, Xin B. Distributed differential evolution algorithm with adaptive parameters[J]. Control and Decision, 2014, 29(4): 701-706.)

[17] Das S, Maity S, Qu B Y, et al. Real-parameter evolutionary multimodal optimization — A survey of the state-of-the-art[J]. Swarm and Evolutionary Computation, 2011, 1(2): 71-88.

[18] Liang J J, Qu B Y, Mao X B, et al. Differential evolution based on fitness Euclidean-distance ratio for multimodal optimization[J]. Neurocomputing, 2014, 137(8): 252-260.

[19] Li X. Niching without niching parameters: particle swarm optimization using a ring topology[J]. IEEE Trans on Evolutionary Computation, 2010, 14(1): 150-169.

[20] García S, Fernández A, Luengo J, et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power[J]. Information Sciences, 2010, 180(10): 2044-2064.

(责任编辑：孙艺红)