计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(3):117−125 [doi: 10.15888/j.cnki.csa.007824]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

多模态优化问题的邻域低密度个体差分进化算法①

闵 涛, 杨 胜

(西安理工大学 理学院应用数学系, 西安 710054)
通讯作者: 闵 涛, E-mail: mintao2003@163.com

摘 要: 针对多模态优化问题 (MultiModal Optimization Problems, MMOPs) 的求解, 提出了一种基于邻域低密度个体的差分进化算法. 该算法在每一代, 首先使用密度峰值聚类的方法求得每一个个体的密度, 然后, 将当前个体邻域范围内密度更低的个体作为变异算子的基向量, 随着种群的进化, 算法将会自动从探索阶段转化为收敛阶段, 进而平衡算法的探索与收敛能力. 将提出的算法应用于 CEC2013 多模态基准测试函数并进行仿真实验, 结果表明本文算法在评价指标峰值比和稳定性上与其它基于差分进化的多模态优化算法相比具有明显的优势, 并随着测试函数的维度与复杂性的增大, 优势就更加明显, 其性能优于许多现有的基于差分进化的多模态优化算法.

关键词: 差分进化; 邻域突变; 多模态; 优化问题; 密度

引用格式: 闵涛,杨胜.多模态优化问题的邻域低密度个体差分进化算法.计算机系统应用,2021,30(3):117–125. http://www.c-s-a.org.cn/1003-3254/7824.html

Differential Evolution Algorithm Based on Low-Density Individual in Neighborhood for Multimodal Optimization Problem

MIN Tao, YANG Sheng

(School of Science, Xi'an University of Technology, Xi'an 710054, China)

Abstract: A differential evolution algorithm based on low-density individuals in the neighborhood is proposed to solve MultiModal Optimization Problems (MMOPs). In each generation, the algorithm first relies on density peak clustering to find the density of each individual and then take the lower-density individuals in the neighborhood of the current individual as a base vector of the mutation operator. As the population evolves, the algorithm will automatically transform from the exploration stage to the convergence stage, thereby balancing its exploration and convergence capabilities. The proposed algorithm is applied to the CEC2013 multimodal benchmark function for simulation experiments. Results demonstrate that the algorithm has obvious advantages over other multimodal optimization algorithms based on differential evolution in evaluating the peak ratios and stability of indexes, and the advantage is more distinct with the increasing dimensionality and complexity of the test function. It behaves better than many existing multimodal optimization algorithms based on differential evolution.

Key words: differential evolution; neighborhood mutation; multimodal; optimization problem; density

科学与工程领域的大多数优化问题本质上都是"多模态"的, 这类问题被称为多模态优化问题, 其特点是目标函数同时具有多个全局最优解或局部最优解. 传统使用的求解方法在求解 MMOPs 时须反复计算多

① 基金项目: 国家自然科学基金 (51679186); 陕西省自然科学基础研究计划 (2019JM-284)
Foundation item: National Natural Science Foundation of China (51679186); Fundamental Research Program of Natural Science of Shaanxi Province (2019JM-284)
收稿时间: 2020-07-10; 修改时间: 2020-08-11; 采用时间: 2020-08-21; csa 在线出版时间: 2021-03-03

次, 才有可能找到不同的最优解, 这样既费时又费力. 因此, 寻找可行有效的求解 MMOPs 的新算法一直以来备受广大科技工作者的关注[1–4]. 为此, 许多被称为小生境的技术包括拥挤[5]、共享[6]、聚类[7]、物种[8] 和种群拓扑[9] 等被提出. 小生境的主要思想是将整个种群划分为几个小生境 (亚种群), 每个亚种群集中于寻找一个或几个最优值.

差分进化算法 (Differential Evolution, DE)[10] 是由 Storn 和 Price 在 1995 年提出的一种全局优化启发式算法. 由于其简单、有效, 已被用于处理各种各样的优化问题, 例如全局优化问题[11], MMOPs[12], 多目标优化问题[13] 等. 近年来,一些基于小生境的 DE 变体[14–18] 被提出以处理 MMOPs, 但是, 这些方法在最优解数量大、维度高和复杂性高的时候效果往往不太理想.

本文提出了一种基于邻域低密度个体的差分进化算法 (Low Density Points Differential Evolutionary, LDPDE) 求解 MMOPs. 算法在每一代, 首先使用密度峰值聚类的方法求得每一个个体的密度, 然后, 优先对邻域范围密度低于当前个体的目标个体进行突变操作, 随着种群的进化, 种群个体逐渐收敛到小生境中心, 进化过程将会自动的从探索阶段转化为收敛阶段, 进而平衡算法的探索与收敛能力. LDPDE 算法的主要特点有: (1) LDPDE 是一种基于距离的小生境方法, 它可以通过邻域范围内个体的密度变化达到算法探索与收敛之间的平衡. (2) 提出了一种新的 DE 变异算子, 称为 DE/low-density/1, 该算子在算法的探索阶段可以寻找尽可能多的有效个体.

1 差分进化算法

考虑优化模型:

$$\min_{x\in\Omega\subseteq R^n} f(X) \tag{1}$$

其中, $X=[x_1,x_2,\cdots,x_n]$为n维向量, $\Omega\in R^n$为求解区域, $f(\cdot)$为目标函数.

差分进化算法是一种简单有效的全局优化启发式算法. 与其他进化算法类似, DE 有 4 个步骤: 初始化、变异、交叉和选择. 标准的 DE 算法过程如下:

(1) 初始化

首先, DE 从最小和最大界限约束的搜索空间内对个体进行均匀随机化, 可以表示为:

$$x_{i,n}^j = x_{\min}^j + rand(0,1)\cdot\left(x_{\max}^j - x_{\min}^j\right)$$

其中, $x_{i,n}^j$表示第 n 代第 i 个个体$x_{i,n}$在第 j 个维度上的值. $x_{\max}^j$和$x_{\min}^j$表示第j个变量的上下边界. 下标 i 和 j 在$[1,NP]$和$[1,D]$范围内, 其中 NP 表示种群大小, D 表示问题维度. $rand(0,1)$表示从 0 到 1 生成均匀分布的随机值.

(2) 变异

初始化后, 从当前种群中随机选择的个体经过变异算子产生突变个体v_i. 在标准 DE 中, 使用以下运算方法:

$$v_i = x_{r_1^i,n} + F\cdot\left(x_{r_2^i,n} - x_{r_3^i,n}\right)$$

其中, r_1^i, r_2^i和r_3^i是从$\{1,2,\cdots,NP\}/\{i\}$中生成的互斥整数, F 是比例因子, 一般取值为 0.1 到 0.9 之间.

(3) 交叉

交叉算子通过重新组合目标个体$x_{i,n}^j$和突变个体v_i来生成实验个体u_i. 在标准 DE 中, 使用了两种类型的算子, 即指数和二项式. 在本文研究中, 我们使用二项式运算符. 在二项式运算符中, 实验个体中的每个元素都继承用户指定的概率为 Cr 的突变个体的值, 并以$(1-Cr)$的概率继承目标个体的值. 实验个体u_i为:

$$u_i^j = \begin{cases} v_i^j, & rand < Cr \vee j = j_{rand} \\ x_{i,n}^j, & \text{otherwise} \end{cases}$$

其中, $j_{rand}\in\{1,2,\cdots,NP\}$是随机选择的索引, 可确保$u_i$至少继承突变个体$v_i$中的至少一个分量.

(4) 选择

选择运算符确定目标或实验个体是否进入到下一代. 如果新的实验个体的函数值等于或小于目标个体的函数值, 它将替换相应的目标个体. 下一代$x_{i,n+1}$中的目标向量为:

$$x_{i,n+1} = \begin{cases} u_i, & f(u_i) \le f(x_{i,n}) \\ x_{i,n}, & f(x_{i,n}) < f(u_i) \end{cases}$$

最后, DE 继续执行变异、交叉和选择运算符, 直到满足终止条件为止. 算法 1 中以伪代码的形式给出了标准 DE 的算法过程.

算法 1. 标准差分进化算法 (DE)

输入: 目标函数 (最小值)$f(x)$; 比例因子 F; 交叉率 Cr; 种群大小 NP; 进化代数 $MaxIt$.

1) 随机初始化种群$X_1=[x_{1,1},x_{2,1},\cdots,x_{NP,1}]$;
2) for $n=1,\cdots,MaxIt$ do
3) for $i=1,\cdots,NP$ do
4) $v_i=x_{r_1^i,n}+F\cdot\left(x_{r_2^i,n}-x_{r_3^i,n}\right)$,
5) $u_i^j=\begin{cases} v_i^j, & rand<Cr\vee j=j_{rand} \\ x_{i,n}^j, & \text{otherwise} \end{cases}$

6) end for
7) for i=1,⋯,NP do
8) $x_{i,n+1}=\begin{cases} u_i, & f(u_i)\geq f(x_{i,n}) \\ x_{i,n}, & f(u_i)<f(x_{i,n}) \end{cases}$
9) end for
10) end for
11) 输出X_{MaxIt}中使得$f(x)$最大的个体.

2 准备工作

2.1 密度峰值聚类

密度峰值聚类[19]是一种简单有效的快速聚类方法, 由 Rodriguez 和 Laio[20]通过发现潜在簇的密度峰值提出. 对于每个数据点$x(i)$, 该算法都会计算其局部密度$\rho(i)$以及与距其最近密度更大点的距离$\sigma(i)$, 由此可以得到每个数据点的孤立性. 以下介绍密度峰值聚类的原理.

数据点$x(i)$的局部密度的定义如下所示:

$$\rho(i)=\sum_{j\in I_S\setminus\{i\}} \mathrm{e}^{-\left(\frac{d_{ij}}{d_c}\right)^2} \tag{2}$$

其中, d_{ij}是第 i 个和第 j 个数据点之间的距离, d_c是截止距离. 对于截止距离d_c, 原始论文没有提供正确设置的有效方法. 在文献 [21] 中, 作者提出了一种基于潜在熵的方法, 可以根据数据域自动设置. 对于一个数据集 $x(i)$, $i=1,2,\cdots,n$, 计算第 i 个数据点$x(i)$的电势 $\psi(i)$:

$$\psi(i)=\sum_{j=1}^{n} \mathrm{e}^{-\left(\frac{\|x(i)-x(j)\|}{\sigma}\right)^2} \tag{3}$$

则熵 H 为:

$$H=-\sum_{i=1}^{n}\frac{\psi(i)}{z}\log\left(\frac{\psi(i)}{z}\right) \tag{4}$$

其中, 归一化因子 z 为:

$$z=\sum_{i=1}^{n}\psi(i) \tag{5}$$

现在只要找到使得 H 值最小的σ, 记为σ', 即可得到截止距离d_c. 显然, d_c的值一定处于$\min\{\|x(i)-x(j)\|\}$和$\max\{\|x(i)-x(j)\|\}$之间, 对于简单的一维优化问题本文使用黄金分割法求得σ, 最后将临界值设置为$d_c=3\cdot\sqrt{2}^{-1}\sigma'$.

2.2 其他多模态优化算法

为了解决多模态优化问题, 许多策略被提出并将嵌入到 DE 中. 以下将简要介绍与 DE 相结合的各种策略.

(1) 拥挤 (CDE): 文献 [14] 将拥挤方案和健身共享嵌入到 DE 中, 分别形成 CDE 算法和 SharingDE 算法. 在 CDE 算法中, 对于每个子代个体, 通过在父代种群中随机选择 C 个父代个体, 然后将子代与该群体中最相似 (欧氏距离最近) 的父代进行比较. 如果子代更好, 它将取代该种群中最相似的父代个体. 否则, 子代个体将被遗弃. SharingDE 算法则利用共享方案来防止种群收敛到同一个高峰.

(2) 物种 (SDE): 在 SDE 算法[15]中, 根据个体的适宜性和小生半径 r 将整个种群分为几个物种. 每个物种都有一个具有最佳适应性的物种种子, 并且 DE 运算符只在当前物种种群内执行. 然后将生成的 N 个子代个体与 N 个父代个体合并为一个种群, 从合并种群中选择 N 个最佳个体, 形成一个新的种群.

(3) 聚类: Qu 等[16]提出了嵌入到 CDE 算法和 SDE 算法中的邻域突变策略 (包括 NCDE 和 NSDE) 中, DE 运算符在每个个体的欧几里得邻域中执行. 遵循这一思想, Gao 等[17]提出了一种具有自适应策略 (包括 self-CCDE 和 CSDE) 的聚类 DE, 它采用自适应参数控制来增强 DE 的搜索能力. Zhang 等[18]提出了一种新的基于位置敏感哈希的小生境技术以降低时间复杂度, 并将其应用于 NCDE 中, 称为 Fast-NCDE 算法.

以上这些方法在处理 MMOPs 问题时已经显示出各方面的有效性, 但它们也有一定的缺点. 一方面, 随着 MMOPs 问题全局最优解的数量增加, 这些方法将全部最优值点找到的机会就越小; 另一方面, 在解决具有高维度和高复杂性的 MMOPs 时, 几乎所有这些固定方法都会失效. 因此, 为解决高维度和高复杂性的 MMOPs, 许多其他基于 DE 的多模态算法被提出. Biswas 等[22]提出了一种新的信息共享机制, 该机制使用本地信息矩阵来诱导有效的利基行为, 称为 LoINDE 算法 (包括 LoICDE 算法和 LoISDE 算法). 同时, 他们还提出了一种新的以父代为中心的标准化邻域 (PCNN) 变异算子, 以维持多个最优值, 称为 PNPCDE 算法[23]. Wang 等[24]提出了是一种基于距离的小生境方法, 称为 MSTDE 算法, 通过 DPR 策略切割少量大的加权边来形成多个小生境, 平衡收敛性和多样性.

3 基于低密度的差分进化算法

在本节中, 我们将提出基于密度峰值的差分进化算法 (LDPDE).

3.1 基于孤立个体的变异算子

受文献 [25] 启发, 提出一种用于多模态优化的新 DE

变异算子, 称为 DE/low-density/1. 该算子的主要特征是它在当前个体邻域中随机选择密度低于当前个体的个体作为变异算子的目标个体, 因此, 目标个体可以被选择变异算子迁移到隔离区域. 在典型的小生境方法中, 同一山谷中的个体倾向于通过在山谷中下降聚集在一起. 而在本文提出的方法中, 个体积极地迁移到其他山谷. DE/low-densit/1 中使用的变异算子描述如下:

$$v_i = x^{\text{near}}_{r_1^p,n} + F_1 \cdot \left(x_{r_2^i,n} - x'_{r_2^i,n}\right) \tag{6}$$

其中, r_1^p 是从$\{p^{\text{near}}(j) < p(i) | j = 1,2,\cdots,N_{d1}\}$中生成随机整数, p^{near}是$x_{i,n}$的N_{d1}近邻个体的密度. r_2^i是从$\{1,2,\cdots, NP\}/\{i, r_1^p\}$中生成随机整数, 差分向量的第二项$x'_{r_2^i,n}$是从$x_{r_2^i,n}$的$N_{d2}$近邻中随机选取, N_{d1}和$N_{d2} \in \{1,2,\cdots,NP-1\}$是用户指定的控制参数. 当$N_{d1}$和$N_{d2}$取值为$NP-1$时, 差分向量的选取方法就与标准 DE 的差分向量取法相同, 只是目标个体不是随机取得. 第二项的设计意图是生成绝对接近孤立个体的实验个体. F_1是比例因子.

3.2 邻域变异算子

当$x_{i,n}$是$x_{i,n}$的N_{d1}邻域内密度最大的个体时, $find(i) = \{p^{\text{near}}(j) < p(i) | j = 1,2,\cdots,N_{d1}\}$为空, 此时上述差分算子将会失效, 这种情况下我们使用如下邻域变异算子来解决困境, 变异运算符如下:

$$v_i = x^{\text{near}}_{r_1^i,n} + F_2 \cdot \left(x'_{r_2^i,n} - x'_{r_3^i,n}\right) \tag{7}$$

其中, $x^{\text{near}}_{r_1^i,n}$是在$x_{i,n}$的$N_{d1}$近邻个体中随机选取的第$r_1^i$个个体, $r_1^i \in \{1,2,\cdots,N_{d1}\}$. $x'_{r_2^i,n}$和$x'_{r_3^i,n}$是从$x_{i,n}$的N_{d3}近邻中随机选取, 且$x^{\text{near}}_{r_1^i,n}$、$x'_{r_2^i,n}$和$x'_{r_3^i,n}$互不相同, $N_{d3} \in \{1,2,\cdots, NP-1\}$是用户指定的控制参数. $N_{d3} = NP-1$的情况与标准 DE 的差分向量取法相同. F_2是比例因子.

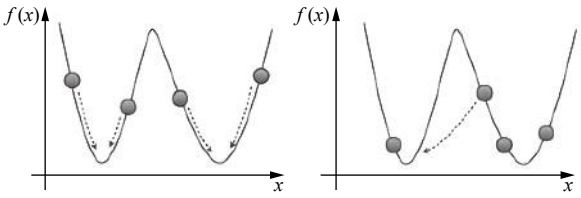
3.3 改进的变异算子

如图 1 所示[25], 图 1(a) 为邻域变异算子的行为特征, 图 1(b) 为基于孤立个体的变异算子的行为特征. 基于孤立个体的变异算子可以促使种群个体向孤立个体附近迁移, 提高算法的多模态开发能力, 而邻域变异算子只在小生境内进行, 避免差分进化的贪婪原则带来的错误替换, 提高算法的收敛性.

将以上两种变异操作结合形成的改进的变异算子描述如下:

$$v_i = \begin{cases} x^{\text{near}}_{r_1^p,n} + F_1 \cdot \left(x_{r_2^i,n} - x'_{r_2^i,n}\right), & find(i)\text{非空} \\ x^{\text{near}}_{r_1^i,n} + F_2 \cdot \left(x'_{r_2^i,n} - x'_{r_3^i,n}\right), & \text{otherwise} \end{cases} \tag{8}$$

随着种群的进化, $\{p^{\text{near}}(j) | j = 1,2,\cdots,N_{d1}\}$的值会趋于相等, $find(i)$为空的情况增多, 基于孤立个体的变异算子的使用率下降, 邻域变异算子的使用率提升, 进而实现算法从探索阶段到收敛阶段的转变, 平衡算法的探索与收敛能力.



(a) 邻域变异算子 (b) 孤立个体变异算子

图 1 两种变异算子的行为特征

LDPDE 的算法过程的伪代码的在算法 2 给出.

算法 2. 基于邻域低密度个体的差分进化算法 (LDPDE)

输入: 目标函数 (最小值)$f(x)$; 比例因子 F; 交叉率 Cr; 种群大小 NP; 最大计算次数 $MaxFES$; 邻域大小N_{d1},N_{d2},N_{d3}

1) 随机初始化种群$X_1=[x_{1,1},x_{2,1},\cdots,x_{NP,1}]$;
2) n=0;
3) while $FES<=MaxFES$ do
4) n=n+1;
5) for i=1,$\cdots$,NP do
6) $d_{i,i}$=0;
7) for j=i+1,$\cdots$,NP do
8) $d_{i,j}=d_{j,i}=\|x_{i,n}-x_{j,n}\|_2$;
9) end for
10) end for
11) $\sigma'=3\cdot\sqrt{2}^{-1}\arg\min_{\sigma} H(\sigma)$;
12) for i=1,$\cdots$,NP do
13) $\rho(i)=\sum_{j\in I_S\setminus\{i\}} \exp\left(-\left(\frac{d_{ij}}{d_c}\right)^2\right)$;
14) end for
15) for i=1,$\cdots$,NP do
16) $v_i=\begin{cases} x^{\text{near}}_{r_1^p,n}+F_1\cdot\left(x_{r_2^i,n}-x'_{r_2^i,n}\right), & find(i)\text{非空} \\ x^{\text{near}}_{r_1^i,n}+F_2\cdot\left(x'_{r_2^i,n}-x'_{r_3^i,n}\right), & \text{otherwise} \end{cases}$
17) $u_i^j=\begin{cases} v_i^j, & rand<Cr \vee j=j_{rand} \\ x_{i,n}^j, & \text{otherwise} \end{cases}$
18) end for
19) for i=1,$\cdots$,NP do
20) $x_{i,n+1}=\begin{cases} u_i, & f(u_i)\geq f(x_{i,n}) \\ x_{i,n}, & f(u_i)<f(x_{i,n}) \end{cases}$
21) end for
22) end for
23) 输出X_{MaxIt}中每个小生境中最优个体.

4 实验与讨论

4.1 测试函数

本文采用了 CEC2013 测试套件[26] 中所有 20 种常

用的多模态测试函数来评估 LDPDE 的性能, 该测试套件中的测试函数具有各种不同的特征, 这使实验更加全面且令人信服. 下面对这 20 个基准测试函数作简要说明.

$F1$: Five-Uneven-Peak Trap(1D), $x \in [0,30]$. 全局最优点数量为 2.

$$F1(x)=\begin{cases}80(2.5-x), & 0\le x<2.5\\ 64(x-2.5), & 2.5\le x<5.0\\ 64(7.5-x), & 5.0\le x<7.5\\ 28(x-7.5), & 7.5\le x<12.5\\ 28(17.5-x), & 12.5\le x<17.5\\ 32(x-17.5), & 17.5\le x<22.5\\ 32(27.5-x), & 22.5\le x<27.5\\ 80(x-27.5), & 27.5\le x\le 30\end{cases}$$

$F2$: Equal Maxima(1D), $x \in [0,1]$. 全局最优点数量为 5.

$$F_2(x)=\sin^6(5\pi x)$$

$F3$: Uneven Decreasing Maxima(1D), $x \in [0,1]$. 全局最优点数量为 1.

$$F3(x)=\mathrm{e}^{-2\log(2)\left(\frac{x-0.08}{0.853}\right)^2}\sin^6\left(5\pi\left(x^{3/4}-0.05\right)\right)$$

$F4$: Himmelblau(2D), $x_1,x_2 \in [-6,6]$. 全局最优点数量为 4.

$$F4(x_1,x_2)=200-\left(x_1^2+x_2-11\right)^2-\left(x_1+x_2^2-7\right)^2$$

$F5$: Six-Hump Camel Back(2D), $x_1 \in [-1.9,1.9]$, $x_2 \in [-1.1,1.1]$. 全局最优点数量为 2.

$$F5(x_1,x_2)=-4\left[\left(4-2.1x_1^2+\frac{x_1^4}{3}\right)x_1^2+x_1x_2+\left(4x_2^2-4\right)x_2^2\right]$$

$F6$, $F8$: Shubert(2D, 3D), $x_i \in [-10,10]^D$, $i=1,2,\cdots,D$. 全局最优点数量分别为 18 和 81.

$$F6,8\left(\vec{x}\right)=-\prod_{i=1}^{D}\sum_{j=1}^{5}j\cos\left[(j+1)x_i+j\right]$$

$F7$, $F9$: Vincent(2D, 3D), $x_i \in [0.25,10]^D$, $i=1,2,\cdots,D$. 全局最优点数量分别为 36 和 216.

$$F7,9\left(\vec{x}\right)=\frac{1}{D}\sum_{i=1}^{D}\sin(10\log(x_i))$$

$F10$: Modified Rastrigin-All Global Optima(2D), $x_i \in [0,1]^D, i=1,2,\cdots,D$. 全局最优点数量为 12.

$$F10\left(\vec{x}\right)=-\sum_{i=1}^{D}(10+9\cos(2\pi k_i x_i))$$

其中, $k_1=3$, $k_2=4$.

剩下的 10 个函数为 4 个复合函数在不同维度大小的情况, 把 4 个函数简称为$CF1,CF2,CF3$和$CF4$, 定义域均为$x_i \in [-5,5]^D, i=1,2,\cdots,D$. 复合函数的具体定义可以参考文献 [26].

$F11$: $CF1$(2D). 全局最优点数量为 6.

$F12$: $CF2$(2D). 全局最优点数量为 8.

$F13$, $F14$, $F16$, $F18$: $CF3$(2D, 3D, 5D, 10D). 全局最优点数量为 6.

$F15$, $F17$, $F19$, $F20$: $CF4$(3D, 5D, 10D, 20 D). 全局最优点数量为 8.

将 LDPDE 与几种基于 DE 的多模态算法进行比较, 包括 CDE 算法, SDE 算法, NCDE 算法, NSDE 算法, LoICDE 算法, LoISDE 算法, PNPCDE 算法, Fast-NCDE 算法和 MSTDE 算法. 这些算法横跨从 2004 年到 2019 年的时间间隔. 此外, 为了具有可比性, 所有对比算法的参数配置都与原始论文中的设置相同.

4.2 参数设置

LDPDE 中的放大系数F_1和F_2分别设置为 0.9 和 0.5, 交叉速率 Cr 设置为 0.9, N_{d2}和N_{d3}分别为 5 和 15. 针对不同的函数参数N_{d1}、种群规模 N 和最大计算次数 $MaxFEs$ 设置如表 1 所示.

表 1 参数设置

函数编号	N_{d1}	N	$MaxFEs$
$F1$–$F5$	5	80	5e+4
$F6$	100	100	2e+5
$F7$	300	300	2e+5
$F8$–$F9$	300	300	4e+5
$F10$	5	100	2e+5
$F11$–$F13$	5	200	2e+5
$F14$–$F17$	5	200	4e+5
$F18$–$F20$	2	200	4e+5

LDPDE 中的种群规模、最大计算次数 $MaxFEs$ 和比较算法的结果参考自文献 [24]. 所有的结果都是在同一测试套件中使用相同的 $MaxFEs$ 下得到的, 每种算法均运行 51 次以进行统计并避免随机性.

对于给定的 $MaxFEs$ 在精度水平$\varepsilon=10^{-4}$下, 使用峰值比 (Peak Ratio, PR) 和成功率 (Success Ratio, SR) 评估算法的性能. 峰值比为找到的全局最优解个数与所有全局最优解数之比, 成功率为所有全局最优解被找到次数与实验总次数之比. PR 和 SR 均为 0 到 1.0 之间的实数, 数值越大说明算法效果越好. 当 PR 和 SR 均为 1.0 时说明算法每次都能找到函数所有的最优解.

4.3 结果与讨论

表 2 列出了 LDPDE 与其他多模态算法在 $F1$–$F20$ 上 PR 和 SR 的详细比较结果. 为了结果清晰, 最佳 PR

果用黑体标出. 符号"+"、"−"和"≈"分别表示 LDPDE 的性能“显著优于”, “显著低于”和“与对比算法相当”. 以 CDE 算法为例, 本文算法在 13 个函数上优于 CDE 算法, 0 个低于 CDE 算法, 7 个与 CDE 算法结果相当.

表 2 LDPDE 与其他多模态算法在 *F*1–*F*20 上 PR 和 SR 的详细比较结果

函数	LDPDE		CDE		SDE		NCDE		NSDE		LoICDE	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
*F*1	**1.000**	1.000	**1.000**	1.000	0.657	0.373	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
*F*2	**1.000**	1.000	**1.000**	1.000	0.737	0.529	**1.000**	1.000	0.776	0.667	**1.000**	1.000
*F*3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
*F*4	**1.000**	1.000	**1.000**	1.000	0.284	0.000	**1.000**	1.000	0.240	0.000	0.975	0.902
*F*5	**1.000**	1.000	**1.000**	1.000	0.922	0.843	**1.000**	1.000	0.745	0.490	**1.000**	1.000
*F*6	**1.000**	1.000	**1.000**	1.000	0.056	0.000	0.305	0.000	0.056	0.000	**1.000**	1.000
*F*7	**1.000**	1.000	0.861	0.000	0.054	0.000	0.873	0.000	0.053	0.000	0.705	0.020
*F*8	**1.000**	1.000	0.000	0.000	0.015	0.000	0.001	0.000	0.013	0.000	0.000	0.000
*F*9	**0.587**	0.000	0.474	0.000	0.011	0.000	0.461	0.000	0.006	0.000	0.187	0.000
*F*10	**1.000**	1.000	**1.000**	1.000	0.147	0.000	0,989	0.863	0.098	0.000	**1.000**	1.000
*F*11	**1.000**	1.000	0.330	0.000	0.314	0.000	0.729	0.000	0.248	0.000	0.660	0.000
*F*12	**0.978**	0.824	0.002	0.000	0.208	0.000	0.252	0.000	0.135	0.000	0.495	0.000
*F*13	**0.954**	0.725	0.141	0.000	0.297	0.000	0.667	0.000	0.225	0.000	0.510	0.000
*F*14	**0.667**	0.000	0.026	0.000	0.216	0.000	**0.667**	0.000	0.190	0.000	0.657	0.000
*F*15	**0.647**	0.000	0.005	0.000	0.108	0.000	0.319	0.000	0.125	0.000	0.299	0.000
*F*16	**0.667**	0.000	0.000	0.000	0.108	0.000	**0.667**	0.000	0.170	0.000	0.559	0.000
*F*17	**0.539**	0.000	0.000	0.000	0.076	0.000	0.250	0.000	0.108	0.000	0.223	0.000
*F*18	**0.578**	0.000	0.167	0.000	0.026	0.000	0.500	0.000	0.163	0.000	0.219	0.000
*F*19	**0.429**	0.000	0.000	0.000	0.105	0.000	0.348	0.000	0.098	0.000	0.037	0.000
*F*20	**0.316**	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.123	0.000	0.123	0.000
	+(显著优于)		13		19		13		18		14	
	−(显著低于)		0		0		0		0		0	
	≈		7		1		7		2		6	

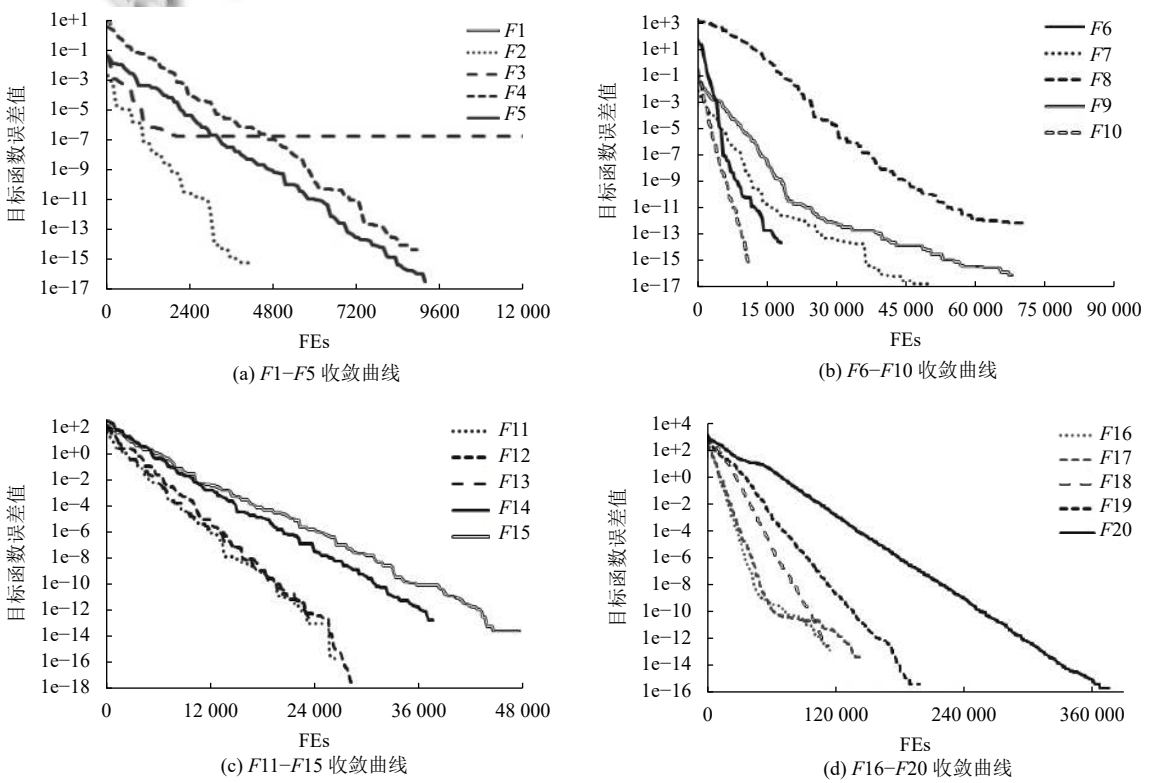
函数	LoISDE		PNPCDE		Self-CCDE		Self-CSDE		Fast-NCDE		MSTDE	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
*F*1	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
*F*2	0.235	0.039	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
*F*3	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000	**1.000**	1.000
*F*4	0.250	0.000	**1.000**	1.000	**1.000**	1.000	0.686	0.294	**1.000**	1.000	**1.000**	1.000
*F*5	0.667	0.333	**1.000**	1.000	**1.000**	1.000	0.961	0.922	**1.000**	1.000	**1.000**	1.000
*F*6	0.056	0.000	0.537	0.000	0.942	0.490	0.699	0.020	0.995	0,902	**1.000**	1.000
*F*7	0.029	0.000	0.874	0.000	0.884	0.020	0.695	0.000	0.675	0.000	0.897	0.000
*F*8	0.012	0.000	0.000	0.000	0.994	0.882	0.695	0.000	0.905	0.000	0.353	0.000
*F*9	0.005	0.000	0.472	0.000	0.459	0.000	0.265	0.000	0.273	0.000	0.491	0.000
*F*10	0.083	0.000	**1.000**	1.000	**1.000**	1.000	0.992	0.922	**1.000**	1.000	**1.000**	1.000
*F*11	0.167	0.000	0.660	0.000	0.778	0.137	0.399	0.000	0.879	0.392	0.667	0.000
*F*12	0.125	0.000	0.000	0.000	0.422	0.000	0.321	0.000	0.956	0.745	0.750	0.000
*F*13	0.167	0.000	0.461	0.000	0.660	0.000	0.317	0.000	0.680	0.000	0.667	0.000
*F*14	0.167	0.000	0.592	0.000	0.657	0.000	0.304	0.000	**0.667**	0.000	**0.667**	0.000
*F*15	0.125	0.000	0.258	0.000	0.343	0.000	0.186	0.000	0.431	0.000	0.527	0.000
*F*16	0.167	0.000	0.000	0.000	0.657	0.000	0.072	0.000	0.660	0.000	**0.667**	0.000
*F*17	0.076	0.000	0.000	0.000	0.248	0.000	0.056	0.000	0.257	0.000	0.316	0.000
*F*18	0.157	0.000	0.147	0.000	0.337	0.000	0.003	0.000	0.412	0.000	0.474	0.000
*F*19	0.027	0.000	0.000	0.000	0.113	0.000	0.000	0.000	0.123	0.000	0.326	0.000
*F*20	0.088	0.000	0.000	0.000	0.027	0.000	0.000	0.000	0.167	0.000	0.123	0.000
+	18		14		14		17		13		11	
−	0		0		0		0		0		0	
≈	2		6		6		3		7		9	

注: 最佳PR结果用黑体标出

从表 2 可以看到 LDPDE 在所有函数上都表现很好. 对于简单函数$F1-F5$和具有众多全局最优值的函数$F6-F10$, LDPDE 除了对于具有 216 个全局最优点的$F9$之外的其它九个函数的PR值全部达到了1.0, 也就是说 LDPDE 算法在这九个测试函数上每次都能找到函数所有的最优解, 虽然$F9$的PR值只有0.587, 但也明显优于于其它算法. 对于最后的十个高度复杂多模态函数$F11-F20$ (其中$F11-F15$为低维, $F16-F20$为高维), LDPDE 的性能几乎处于绝对的优势地位, 只有少数几个算法在一到两个函数可以与本文算法性能相当, 这进一步验证了 LDPDE 在解决高度复杂的 MMOPs 方面的可行性和有效性.

多模态函数$F1-F20$的收敛曲线如图 2(a)–图 2(d)所示, 目标函数误差值为每个函数运行 51 次所得每一代最小值的平均值与函数极值之差的绝对值, 为方便观察, 将纵坐标取以10为底的对数 (值为0时曲线未画出). 图 2(a) 中函数$F1$的曲线极短, 这是由于$F1$函数简单, 算法在很少的进化代数就计算得出最优函数值导致的. 而函数$F3$的计算精度达到10^{-7}之后很难继续改进. 其他 18 个函数收敛曲线呈现明显的收敛趋势, 虽然随着函数复杂程度的提高收敛速度减慢, 但收敛趋势稳定, 收敛精度也都能达到10^{-18}, 这也体现出本文算法在高维复杂多模态函数优化上的优势.

总之, LDPDE 在所有 20 个函数上与 CDE, SDE, NCDE, NSDE, LIPS, LoICDE, LoISDE, PNPCDE, self-CCDE, self-CSDE, fast-NCDE 和 MSTDE 算法相比PR值都优于或相当于对比算法. 收敛性分析结果表明, LDPDE 可以在搜索与收敛之间找到良好的平衡, 具有解决 MMOPs 的潜力.



(a) $F1$–$F5$ 收敛曲线 (b) $F6$–$F10$ 收敛曲线

(c) $F11$–$F15$ 收敛曲线 (d) $F16$–$F20$ 收敛曲线

图 2 目标函数误差值收敛曲线

5 结论与展望

针对多模态优化问题, 本文提出了一种基于邻域低密度个体的差分进化算法 LDPDE, 它可以通过邻域范围内个体的密度变化达到算法探索与收敛之间的平衡, 有效提升了算法的可靠性和收敛速度, 结合新提出的 DE/low-density/1 变异算子, 使得算法在探索阶段可

以寻找尽可能多的有效个体. 在 CEC2013 多模态基准测试的实验结果表明, LDPDE 对比其它多种基于 DE 的多模态优化算法具有更好的性能, 显示了 LDPDE 在求解多模态优化问题上的优势.

参考文献

1 Pham MT, Song MH, Koh CS. Coupling particles swarm optimization for multimodal electromagnetic problems. Journal of Electrical Engineering & Technology, 2010, 5(3): 423–430. [doi: 10.5370/JEET.2010.5.3.423]

2 Zhao SZ, Suganthan PN. Diversity enhanced particle swarm optimizer for global optimization of multimodal problems. Proceedings of 2009 IEEE Congress on Evolutionary Computation. Trondheim, Norway. 2009. 590–597. [doi: 10.1109/CEC.2009.4982999]

3 Kumar V, Chhabra JK, Kumar D. Variance-based harmony search algorithm for unimodal and multimodal optimization problems with application to clustering. Cybernetics and Systems, 2014, 45(6): 486–511. [doi: 10.1080/01969722.2014.929349]

4 Lapizco-Encinas G, Kingsford C, Reggia J. Particle swarm optimization for multimodal combinatorial problems and its application to protein design. Proceedings of 2010 IEEE Congress on Evolutionary Computation. Barcelona, Spain. 2010. 1–8. [doi: 10.1109/CEC.2010.5586157]

5 De Jong K. Analysis of the behavior of a class of genetic adaptive systems[Ph.D. thesis]. Ann Arbor: University of Michigan, 1975.

6 Lin CY, Wu WH. Niche identification techniques in multimodal genetic search with sharing scheme. Advances in Engineering Software, 2002, 33(11–12): 779–791. [doi: 10.1016/S0965-9978(02)00045-5]

7 Yin XD, Germay N. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In: Albrecht RF, Reeves CR, Steele NC, eds. Artificial Neural Nets and Genetic Algorithms. Vienna: Springer, 1993. 450–457. [doi: 10.1007/978-3-7091-7533-0_65]

8 Bessaou M, Pétrowski A, Siarry P. Island model cooperating with speciation for multimodal optimization. Proceedings of the 6th International Conference on Parallel Problem Solving from Nature PPSN VI. Paris, France. 2000. 437–446. [doi: 10.1007/3-540-45356-3_43]

9 Kennedy J, Mendes R. Population structure and particle swarm performance. Proceedings of 2002 Congress on Evolutionary Computation. Honolulu, HI, USA. 2002. 1671–1676. [doi: 10.1109/CEC.2002.1004493]

10 Storn R, Price K. Differential evolution —A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341–359. [doi: 10.1023/A:1008202821328]

11 Zhou XG, Zhang GJ, Hao XH, *et al*. A novel differential evolution algorithm using local abstract convex underestimate strategy for global optimization. Computers & Operations Research, 2016, 75: 132–149. [doi: 10.1016/j.cor.2016.05.015]

12 Wang HF, Moon I, Yang SX, *et al*. A memetic particle swarm optimization algorithm for multimodal optimization problems. Information Sciences, 2012, 197: 38–52. [doi: 10.1016/j.ins.2012.02.016]

13 Zhang ZH, Zhong CQ, Xu ZZ, *et al*. A non-dominated sorting cooperative co-evolutionary differential evolution algorithm for multi-objective layout optimization. IEEE Access, 2017, 5: 14468–14477. [doi: 10.1109/ACCESS.2017.2716111]

14 Thomsen R. Multimodal optimization using crowding-based differential evolution. Proceedings of 2004 Congress on Evolutionary Computation. Portland, OR, USA. 2004. 1382–1389. [doi: 10.1109/CEC.2004.1331058]

15 Li XD. Efficient differential evolution using speciation for multimodal function optimization. Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. Washington, DC, USA. 2005. 873–880. [doi: 10.1145/1068009.1068156]

16 Qu BY, Suganthan PN, Liang JJ. Differential evolution with neighborhood mutation for multimodal optimization. IEEE Transactions on Evolutionary Computation, 2012, 16(5): 601–614. [doi: 10.1109/TEVC.2011.2161873]

17 Gao WF, Yen GG, Liu SY. A cluster-based differential evolution with self-adaptive strategy for multimodal optimization. IEEE Transactions on Cybernetics, 2014, 44(8): 1314–1327. [doi: 10.1109/TCYB.2013.2282491]

18 Zhang YH, Gong YJ, Zhang HX, *et al*. Toward fast niching evolutionary algorithms: A locality sensitive hashing-based approach. IEEE Transactions on Evolutionary Computation, 2017, 21(3): 347–362. [doi: 10.1109/TEVC.2016.2604362]

19 邓然然, 李伟, 杨荣新. 自调节步长果蝇优化的自适应密度峰值聚类. 计算机系统应用, 2020, 29(4): 126–136. [doi:

10.15888/j.cnki.csa.007343]

20 Rodriguez A, Laio A. Clustering by fast search and find of density peaks. Science, 2014, 344(6191): 1492–1496. [doi: 10.1126/science.1242072]

21 Wang SL, Wang DK, Li CY, *et al*. Comment on “clustering by fast search and find of density peaks”. arXiv: 1501.04267, 2015.

22 Biswas S, Kundu S, Das S. Inducing niching behavior in differential evolution through local information sharing. IEEE Transactions on Evolutionary Computation, 2015, 19(2): 246–263. [doi: 10.1109/TEVC.2014.2313659]

23 Biswas S, Kundu S, Das S. An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution. IEEE Transactions on Cybernetics, 2014, 44(10): 1726 –1737. [doi: 10.1109/TCYB.2013.2292971]

24 Wang ZJ, Zhan ZH, Zhang J. Distributed minimum spanning tree differential evolution for multimodal optimization problems. Soft Computing, 2019, 23(24): 13339–13349. [doi: 10.1007/s00500-019-03875-x]

25 Otani T, Suzuki R, Arita T. DE/isolated/1: A new mutation operator for multimodal optimization with differential evolution. Proceedings of the 24th International Conference on Advances in Artificial Intelligence. Perth, Australia. 2013. 99–105. [doi: 10.1007/978-3-642-25832-9_33]

26 Li XD, Engelbrecht A, Epitropakis MG. Benchmark functions for CEC’2013 special session and competition on niching methods for multimodal function optimization [Technical Report]. Australia: Evolutionary Computation and Machine Learning Group, RMIT University, 2013.