# 局部抽象凸区域剖分差分进化算法

周晓根 [1] 张贵军 [1] 郝小虎 [1]

**摘 要** 在差分进化算法框架下, 结合抽象凸理论, 提出一种局部抽象凸区域剖分差分进化算法 (Local partition based differential evolution, LPDE). 首先, 通过对新个体的邻近个体构建分段线性下界支撑面, 实现搜索区域的动态剖分; 然后, 利用区域剖分特性逐步缩小搜索空间, 同时根据下界估计信息指导种群更新, 并筛选出较差个体; 其次, 借助下界支撑面的广义下降方向作局部增强, 并根据进化信息对搜索区域进行二次剖分; 最后, 根据个体的局部邻域下降方向对部分较差个体作增强处理. 数值实验结果表明了所提算法的有效性.

**关键词** 差分进化, 区域剖分, 全局优化, 下界估计, 抽象凸

**引用格式** 周晓根, 张贵军, 郝小虎. 局部抽象凸区域剖分差分进化算法. 自动化学报, 2015, **41**(7): 1315−1327

**DOI** 10.16383/j.aas.2015.c140680

## Differential Evolution Algorithm with Local Abstract Convex Region Partition

ZHOU Xiao-Gen[1] ZHANG Gui-Jun[1] HAO Xiao-Hu[1]

**Abstract** Within the framework of differential evolution algorithm, a differential evolution algorithm with local abstract convex region partition is proposed in this paper incorporating the abstract convexity theory. Firstly, the partition of the search domain is performed dynamically by building piecewise linear abstract convex lower supporting hyperplanes for the neighboring individuals of new individuals. Secondly, the search domain narrows gradually by using properties of the region partition. Meanwhile, the process of population updating is guided according to the information of underestimate, and poor individuals are identified effectively. Additionally, the generalized descent directions of the lower supporting hyperpalens are used for local enhancement, and the search domain is partitioned again according to the evolutionary information. Finally, some poor individuals get enhanced according to descent directions of their local neighbourhood. Numerical experiment results have verified the effectiveness of the proposed algorithm.

**Key words** Differential evolution, region partition, global optimization, underestimate, abstract convex

**Citation** Zhou Xiao-Gen, Zhang Gui-Jun, Hao Xiao-Hu. Differential evolution algorithm with local abstract convex region partition. *Acta Automatica Sinica*, 2015, **41**(7): 1315−1327

在科学、经济和工程中, 存在着许多复杂的多模连续函数全局优化问题, 如计算科学中的旅行商问题, 生物信息学中的蛋白质结构预测问题和工程中的特殊形状设计问题等, 这些问题在限定的解向量空间可能存在多个全局最优解和大量的局部最优解. 基于梯度的共轭梯度法、拟牛顿法等传统算法[1] 以及 Hooke-Jeeves 模式搜索法[2]、Nelder-Mead 单纯形法[3] 等直接搜索算法本质上都属于一类局部搜索算法, 这些算法不仅对函数的性态有要求, 而且通过这些算法得到的解的质量往往跟初始点的选择有很大关系. 因此, 对于一些复杂的实际优化问题, 通过这些算法难以求得问题的全局最优解.

确定性算法通过构建不断收紧的下界凸包络来逼近目标函数, 从而求得原问题的全局最优解. $\alpha$ 分支定界算法 ($\alpha$-based branch and bound, $\alpha$BB)[4−6] 及割角法 (Cutting angle method, CAM)[7−9] 等典型确定性算法在现有文献中被广泛研究[10], 然而, 就 $\alpha$BB 而言, 其 $\alpha$ 的求解是一个极其富有挑战性的工作[7]; 对 CAM 而言, 为了获得理想的精度, 算法需要构建大量的支撑向量来逼近目标函数, 并需要极大的空间来保存各下界估计值, 随着问题规模的增大, 急剧增长的计算复杂度和空间复杂度限制了其在大规模问题中的应用[10].

近年来, 进化算法等一类随机性算法以简单而高效的优势被广泛应用于电力系统、化学工程以及

收稿日期 2014-12-03 录用日期 2015-02-27
Manuscript received December 3, 2014; accepted February 27, 2015
国家自然科学基金 (61075062), 浙江省自然科学基金 (LY13F030008), 浙江省科技厅公益项目 (2014C33088), 浙江省重中之重学科开放基金 (20120811), 杭州市产学研合作项目 (20131631E31) 资助
Supported by National Natural Science Foundation of China (61075062), Natural Science Foundation of Zhejiang Province (LY13F030008), Public Welfare Project of Science Technology Department of Zhejiang Province (2014C33088), Open Fund for Key-Key Discipline of Zhejiang Province (20120811), and Co-operation Project of Industry-Academia-Research Institute of Hangzhou (20131631E31)
本文责任编委 赵千川
Recommended by Associate Editor ZHAO Qian-Chuan
1. 浙江工业大学信息工程学院 杭州 310023
1. College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023

生物信息学等领域[11−12]. Storn 等提出的差分进化算法 (Different evolution, DE)[13] 通过模仿生物群体内个体间的合作与竞争产生的群体智能实现对优化问题的求解, 具有能够记忆个体最优解, 种群内信息共享及易与其他算法结合的特点, 在各种问题的应用求解中展现出了其独特的优势[14−15], 但在理论和应用中也暴露出诸多不足和缺陷: 1) 贪婪选择策略加快了算法的收敛速度, 但是也使其极易陷入局部最优而降低可靠性; 2) 求解时需要大量的函数评价次数而导致计算代价较高; 3) 全局探测能力较强, 但是局部搜索能力较弱, 后期收敛速度较慢[16].

针对上述问题, 国内外学者通过改变 DE 算法的新个体生成策略, 引入参数和策略自适应机制及与其他算法混合等方式提出了一些改进算法. Kaelo 等[17] 通过在 DE 算法的变异过程加入锦标赛机制提出一种改进 DE 算法 (Differential evolution with random localization, DERL), 同时在选择环节引入反射和收缩算子提出另一种改进 DE 算法 (Differential evolution with localization using the best vector, DELB), 从而降低算法的计算代价; Cai 等[18] 提出一种基于近邻信息和方向信息的差分进化算法 (Neighborhood and direction information based differential evolution, NDi-DE), 在变异过程中, 利用个体的近邻信息选择父代个体, 同时利用一种包含方向的变异策略自适应的获取近邻个体的方向信息, 从而在搜索过程中, 不仅能够快速地定位极小值区域, 加快算法的收敛速度, 而且能够防止个体落入无效区域. Bhattacharya 等[19] 提出一种基于生物地理学优化算法 (Biogeography-based optimization, BBO) 的混合 DE 算法 (DE/BBO), 利用 BBO 算法的迁移操作来指导 DE 算法变异产生新个体, 即对较优个体进行保存, 从而通过较差个体接受较优个体的新特性的方式使得当前种群得到充分探测, 以提高算法的全局搜索能力, 同时加快算法的收敛速度; Wang 等[20] 提出一种具有复合新个体生成策略和控制参数的差分进化算法 (Composite differential evolution, CoDE), 在算法中分别设置一个策略池和一个参数池, 通过策略池中不同的生成策略与参数池中不同的控制参数随机组合来竞争产生新个体, 从而改善 DE 算法的性能; Gong 等[21] 在 DE 算法的变异环节引入排名机制, 提出一种 Rank-DE 算法，在变异过程中, 根据当前种群个体的适应度排名选择父代个体, 从而提高算法的局部搜索能力, 加快算法的收敛速度. 上述改进算法取得了一定的效果, 但是对于实际应用中的一些大规模优化问题, 由于其目标函数曲面极其粗糙复杂, 因此, 对于上述改进算法, 计算代价和收敛速度仍然是算法的瓶颈所在, 而且也极容易出现早熟收敛.

为了提高 DE 算法在计算代价、收敛速度及可靠性方面的性能, 本文在基于广义凸的多模态差分进化算法[22] 和基于局部抽象凸支撑面的多模态优化算法[23] 等基础上, 进一步引入搜索区域动态剖分思想和个体选择性增强策略, 提出一种局部抽象凸区域剖分差分进化算法 (Local partition based differential evolution, LPDE). LPDE 算法利用抽象凸分段线性估计特性, 对新个体的邻近个体构建分段线性下界支撑面, 从而对搜索区域进行动态剖分, 进而利用各区域及其所对应的下界估计区域的剖分特性, 有效地识别出部分无效区域, 逐步缩小搜索区域, 以降低算法的计算代价, 提高算法的搜索效率和可靠性; 其次, 利用下界支撑面估计目标函数值来指导种群更新, 并根据下界估计信息筛选出较差个体, 有效地减少目标函数评价次数; 同时, 借助分段线性下界支撑面的广义下降方向作局部增强, 以加快算法的收敛速度; 另外, 根据进化信息对搜索区域进行再次剖分, 进一步缩小搜索区域; 最后, 根据个体的局部邻域下降方向对部分较差个体作增强处理, 进一步降低算法的计算代价. 设计实现了进化算法与抽象凸方法的协同优化框架.

## 1 区域剖分基本理论

### 1.1 基本定义

**定义 1.** 若存在函数族 $U \subseteq H$, 使得函数 $f: \mathbf{R}_+^N \to \mathbf{R}$ 满足:

$$f(\boldsymbol{x}) = \sup\{h(\boldsymbol{x}) : h \subset U\}, \qquad \forall \boldsymbol{x} \in \mathbf{R}_+^N \tag{1}$$

其中, $H$ 为定义在可行域 $\mathbf{R}_+^N$ 上的函数族, 则称函数 $f$ 是关于函数族 $H$ 的抽象凸函数 ($H$-convex).

**定义 2.** 设 $H$ 为定义在可行域 $\mathbf{R}_+^N$ 上的函数族, 则称

$$\partial_H f(\boldsymbol{x}) = \{h \in H : h(\boldsymbol{y}) \le f(\boldsymbol{y}), \\ f(\boldsymbol{x}) = h(\boldsymbol{x})\}, \quad \forall \boldsymbol{y} \in \mathbf{R}_+^N \tag{2}$$

为函数 $f: \mathbf{R}_+^N \to \mathbf{R}$ 在点 $\boldsymbol{x}$ 处的 $H$-次微分 ($H$-subgradients).

**定义 3.** 若函数 $f$ 满足如下两个条件:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}_+^N, \quad \boldsymbol{x} \ge \boldsymbol{y} \Rightarrow f(\boldsymbol{x}) \ge f(\boldsymbol{y}) \tag{3}$$

$$\forall \boldsymbol{x} \in \mathbf{R}_+^N, \quad \lambda \in \mathbf{R}_{++} \Rightarrow f(\lambda \boldsymbol{x}) = \lambda f(\boldsymbol{x}) \tag{4}$$

则称函数 $f: \mathbf{R}_+^N \to \mathbf{R}$ 为正齐次递增函数 (Increasing positively homogeneous functions of degree one, IPH).

**定义 4.** 若函数 $f: \boldsymbol{x} \to \mathbf{R}$ 对于任意的 $\boldsymbol{x}_1$, $\boldsymbol{x}_2 \in [\boldsymbol{a}, \boldsymbol{b}]$, 都满足:

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \le C||\boldsymbol{x}_1 - \boldsymbol{x}_2|| \tag{5}$$

则称函数 $f$ 关于 $[\boldsymbol{a},\boldsymbol{b}]$ Lipschitz 连续, 其中, $C$ 称为 Lipschitz 常数.

**定义 5.** 针对个体 $\boldsymbol{x}^k$ 有:

$$d_i^k = x_{\mathrm{lbest},i}^{\psi} - x_i^k, \quad i = 1, 2, \cdots, N \tag{6}$$

则称方向 $\boldsymbol{d}^k = (d_1^k, d_2^k, \cdots, d_N^k)$ 为 $\boldsymbol{x}^k$ 的局部邻域下降方向, 其中, $\psi$ 为 $[0.5N_P, N_P]$ 之间的随机整数, $N_P$ 为种群规模, $x_{\mathrm{lbest}}^{\psi}$ 为 $\psi$ 个个体中的最优个体.

### 1.2 区域分段线性下界估计松弛模型

设如下全局优化问题

$$\min\ f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega \subset \mathbf{R}^N \tag{7}$$

满足定义 4, 其中, $f(\cdot)$ 为定义在可行域 $\Omega$ 上的目标函数, $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)^{\mathrm{T}}$ 为其 $N$ 维连续优化变量.

通过如下公式对原优化变量 $\boldsymbol{x} = (x_1, x_2, \cdots, x_N)^{\mathrm{T}}$ 作线性变换:

$$\begin{cases} x'_1 \equiv \dfrac{1}{Z} \\ x'_2 \equiv x'_1 \mathrm{e}^{x_1} \\ x'_3 \equiv x'_1 \mathrm{e}^{x_1+x_2} \\ \quad \vdots \\ x'_{N+1} \equiv x'_1 \mathrm{e}^{\sum\limits_{i=1}^{N} x_i} \end{cases}, \quad i = 1, 2, \cdots, N \tag{8}$$

其中, $Z = 1 + \sum_{i=1}^{N} (\mathrm{e}^{\sum_{j=1}^{i} x_j})$, $\boldsymbol{x}' = (x'_1, x'_2, \cdots, x'_{N+1})^{\mathrm{T}}$ 为定义在 $N+1$ 维单位单纯形区域 $S \equiv \{\boldsymbol{x}' \in \mathbf{R}^{N+1},\ x'_i \geq 0, \sum_{i=1}^{N+1} x'_i = 1\}$ 中的线性转换变量.

由式 (8) 的反变换有:

$$x_i = \frac{\ln x'_{i+1}}{\ln x'_i}, \quad i = 1, 2, \cdots, N \tag{9}$$

将式 (9) 代入式 (7), 则原目标问题转换为

$$\min\ \bar{f}(\boldsymbol{x}'), \quad \boldsymbol{x}' \in S \subset \mathbf{R}_+^{N+1} \tag{10}$$

其中, $\bar{f}(\cdot)$ 为定义在 $N+1$ 维单位单纯形区域 $S$ 上的目标函数. 由文献 [24] 可知, 式 (10) 目标函数可以通过加上常数 $M$ ($M \geq 2C$) 转化为 IPH 函数, 且目标函数的最优值不变, 则原目标问题进一步转换为

$$\min\ g(\boldsymbol{x}') = \bar{f}(\boldsymbol{x}') + M, \quad \boldsymbol{x}' \in S \subset \mathbf{R}_+^{N+1} \tag{11}$$

假定有 $K$ 个式 (11) 的给定点 $(\boldsymbol{x}'^k,\ g(\boldsymbol{x}'^k))$, $k = 1, 2, \cdots, K$, 令 $I = \{1, 2, \cdots, N+1\}$, 则根据各点处的 $H$-次微分可知, 目标函数 (11) 是关于以下函数 (支撑函数) 的抽象凸函数 ($H$-convex):

$$h^k(\boldsymbol{x}') = \min_{i \in I} \frac{x'_i}{l_i}, \quad l \in \mathbf{R}_+^{N+1} \tag{12}$$

由于任一抽象凸函数都可以通过它的一系列简单弱函数的上确界表示[24], 从而可以利用式 (11) 目标函数的一系列支撑函数 $h^k$, $k = 1, 2, \cdots, K$, 建立原目标函数的区域分段线性下界估计松弛模型:

$$H^K(\boldsymbol{x}') = \max_{k \leq K} h^k(\boldsymbol{x}') = \max_{k \leq K} \min_{i \in I} \frac{x'_i}{l_i^k} \tag{13}$$

其中

$$\boldsymbol{l}^k = \left( \frac{x'^k_{\ 1}}{g(x'^k)}, \frac{x'^k_{\ 2}}{g(x'^k)}, \cdots, \frac{x'^k_{\ N+1}}{g(x'^k)} \right) \tag{14}$$

为点 ($\boldsymbol{x}'^k$, $g(\boldsymbol{x}'^k)$) 处的支撑向量.

**引理 1.** 设 $\exists \bar{C} > 0$, 使得式 (10) 的目标函数 $\bar{f} : S \to \mathbf{R}$ 满足:

$$\bar{C} = \inf_{\boldsymbol{x}'^1 \neq \boldsymbol{x}'^2} \frac{|\bar{f}(\boldsymbol{x}'^1) - f(\boldsymbol{x}'^2)|}{||\boldsymbol{x}'^1 - \boldsymbol{x}'^2||_1}, \quad \forall \boldsymbol{x}'^1, \boldsymbol{x}'^2 \in S \tag{15}$$

取 $M > 2\bar{C} - \min_{\boldsymbol{x}' \in S} \bar{f}(\boldsymbol{x}')$, 则 $\forall \boldsymbol{y} \in S$ 处生成的支撑函数 $h^y(\boldsymbol{x}')$ 满足:

$$h^y(\boldsymbol{x}') \leq g(\boldsymbol{x}') = \bar{f}(\boldsymbol{x}') + M, \quad \forall \boldsymbol{x}' \in S \tag{16}$$

其中, $\|\boldsymbol{x}'^1 - \boldsymbol{x}'^2\|_1 \equiv \max_{i \in I} \|\boldsymbol{x}'^1 - \boldsymbol{x}'^2\|$, $\boldsymbol{l} = \boldsymbol{y}/g(\boldsymbol{y})$, $\boldsymbol{l} = (l_1, l_2, \cdots, l_{N+1}) \in \mathbf{R}_+^{N+1}$.

**证明.** 详细证明过程可参见文献 [22]. □

**注 1.** 引理 1 表明, 对于目标函数 (10), 通过加上合适的常数 $M$ 转化为式 (11) 的 IPH 函数后, 根据可行域的采样点 $\boldsymbol{y}$ ($\boldsymbol{y} \in S$) 构建的基于条件 (15) 的下界估计函数 $h^y(\boldsymbol{x}')$ 满足 $f(\boldsymbol{x}') \geq h^y(\boldsymbol{x}') - M, \forall \boldsymbol{x}' \in S$.

**推论 1.** 设 $\boldsymbol{y}^1, \boldsymbol{y}^2, \cdots, \boldsymbol{y}^K \in S$ 为式 (11) 目标函数的已知采样点, 则式 (13) 区域分段线性下界估计松弛模型满足:

$$\begin{aligned} &g(\boldsymbol{x}') \geq H^K(\boldsymbol{x}'), \quad \forall \boldsymbol{x}' \in S \\ &g(\boldsymbol{x}') = H^K(\boldsymbol{x}'), \quad \forall \boldsymbol{x}' \in \{\boldsymbol{y}^1, \cdots, \boldsymbol{y}^K\} \end{aligned}$$

其中, $l_i^k = y_i^k / g(\boldsymbol{y}^k)$, 记 $h^k(\boldsymbol{x}') \equiv h^{y^k}(\boldsymbol{x}'), k = 1, 2, \cdots, K$.

**证明.** 根据引理 1 可知:

$$g(\boldsymbol{x}') \geq h^{\boldsymbol{y}}(\boldsymbol{x}'), \quad \boldsymbol{y} \in S,\ \forall \boldsymbol{x}' \in S \tag{17}$$

故可得到:

$$g(\boldsymbol{x}') \geq \max_{k \leq K} h^k(\boldsymbol{x}') = H^K(\boldsymbol{x}'), \quad \forall \boldsymbol{x}' \in S \tag{18}$$

设 $\boldsymbol{x}' = \boldsymbol{y}^{\xi}, \xi \in \{1, 2, \cdots, K\}$, 则 $g(\boldsymbol{y}^{\xi}) = h^{\xi}(\boldsymbol{y}^{\xi})$, 又由式 (18) 可知:

$$h^{\xi}(\boldsymbol{y}^{\xi}) = H^{k}(\boldsymbol{y}^{\xi})$$

故 $g(\boldsymbol{x}') = H^{K}(\boldsymbol{x}'), \forall \boldsymbol{x}' \in \{y^1, \cdots, y^K\}$. □

**注 2.** 推论 1 说明式 (11) 目标函数的下界估计松弛模型位于目标函数曲线 (面) 以下, 且采样点处的下界估计值等于原目标函数值, 即目标函数的下界估计值小于等于原目标函数值.

### 1.3 区域剖分特性

**定理 1.** 式 (13) 区域分段线性下界估计松弛模型函数为 IPH 函数, 其中 $K \geq N + 1$.

**证明.**

1) 设 $\alpha > 0$, 则:

$$H^{K}(\alpha \boldsymbol{x}') = \max_{k \leq K} \min_{i \in I} \frac{(\alpha x_i')}{l_i^k} = \alpha \max_{k \leq K} \min_{i \in I} \frac{x_i'}{l_i^k} = \alpha H^{k}(\boldsymbol{x}') \tag{19}$$

2) 设 $\boldsymbol{y} \geq \boldsymbol{x}'$, 则:

$$H^{k}(\boldsymbol{y}) - H^{k}(\boldsymbol{x}') = \max_{k \leq K} \min_{i \in I} \frac{y_i}{l_i^k} - \max_{k \leq K} \min_{i \in I} \frac{x_i'}{l_i^k} = \frac{y_{\xi_2}}{l_{\xi_2}^k} - \frac{x_{\xi_1}'}{l_{\xi_1}^k} \tag{20}$$

a) 当 $\xi_2 \neq \xi_1$ 时:

$$\frac{x_{\xi_1}'}{l_{\xi_1}^k} \leq \frac{x_{\xi_2}'}{l_{\xi_2}^k} \leq \frac{y_{\xi_2}}{l_{\xi_2}^k} \tag{21}$$

则 $H^{K}(\boldsymbol{y}) \geq H^{K}(\boldsymbol{x}')$;

b) 当 $\xi_2 = \xi_1$ 时:

$$\frac{x_{\xi_1}'}{l_{\xi_1}^k} = \frac{x_{\xi_2}'}{l_{\xi_2}^k} \leq \frac{y_{\xi_2}}{l_{\xi_2}^k} \tag{22}$$

则 $H^{K}(\boldsymbol{y}) \geq H^{K}(\boldsymbol{x}')$. □

由定理 1 可知, 下界估计松弛模型函数 (13) 为单位单纯形约束下的 IPH 函数, 则各剖分区域及其对应的下界估计区域可由 $N + 1$ 个支撑向量 $\boldsymbol{l}^{k_1}$, $\boldsymbol{l}^{k_2}, \cdots, \boldsymbol{l}^{k_{N+1}}$ 组成的支撑矩阵 $L$ 表示[25]:

$$L = \begin{bmatrix} l_1^{k_1} & l_2^{k_1} & \cdots & l_{N+1}^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \cdots & l_{N+1}^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_{N+1}} & l_2^{k_{N+1}} & \cdots & l_{N+1}^{k_{N+1}} \end{bmatrix} \tag{23}$$

假设 $L$ 满足以下两个条件:

$$\forall i, j \in I, i \neq j : l_i^{k_i} > l_i^{k_j} \tag{24}$$

$$\forall v \in \Lambda^{K} \backslash L, \exists i \in I : l_i^{k_i} \leq v_i \tag{25}$$

则 $L$ 对应的剖分区域具有以下特性:

$$\boldsymbol{x}'_{\min}(L) = \frac{\mathrm{diag}\{L\}}{\mathrm{tr}(L)} \tag{26}$$

$$d(L) = H^{K}(\boldsymbol{x}'_{\min}) = \frac{1}{\mathrm{tr}(L)} \tag{27}$$

其中, $\boldsymbol{x}'_{\min}$ 和 $d$ 分别表示 $L$ 对应的下界估计区域的极值解和极值.

设满足式 (24) 和 (25) 的支撑矩阵 $L^u$ 对应的下界估计区域的极值解为 $\boldsymbol{x}'^{u}$, 由式 (26) 有:

$$x'^{u}_{\ i} = \frac{l_i^{k_i}}{\mathrm{tr}(L^u)} = \frac{x'^{k_i}_{\ i}}{g(\boldsymbol{x}'^{k_i})\mathrm{tr}(L^u)} \tag{28}$$

由式 (24) 有:

$$\frac{x'^{k_i}_{\ i}}{g(\boldsymbol{x}'^{k_i})} > \frac{x'^{k_j}_{\ i}}{g(\boldsymbol{x}'^{k_j})} \tag{29}$$

对式 (29) 两边同时乘以 $x'^{k_j}_{\ j}/\mathrm{tr}(L^u)$:

$$\frac{x'^{k_j}_{\ j}}{\mathrm{tr}(L^u)} \frac{x'^{k_i}_{\ i}}{g(\boldsymbol{x}'^{k_j})} > \frac{x'^{k_j}_{\ i}}{\mathrm{tr}(L^u)} \frac{x'^{k_j}_{\ j}}{g(\boldsymbol{x}'^{k_j})} \tag{30}$$

由式 (28) 和 (30) 有:

$$x'^{k_j}_{\ j} x'^{u}_{\ i} > x'^{k_j}_{\ i} x'^{u}_{\ j} \tag{31}$$

因此, 对于所有的 $i, j \in I$ 有 $N(N+1)$ 个式 (31) 的不等式, 即

$$x'^{k_j}_{\ j} x'^{u}_{\ i} > x'^{k_j}_{\ i} x'^{u}_{\ j}, \quad i, j \in I,\ i \neq j \tag{32}$$

设满足不等式 (32) 的集合为 $S^u$, 则构建抽象凸下界支撑向量形成的区域为 $S^u$, 即支撑矩阵 $L^u$ 对应的区域为 $S^u$, 显然 $S^u \subset S$. 利用下界估计区域的特性 $H^{K}(\boldsymbol{x}'^{u}) \leq H^{K}(\boldsymbol{x}'), \forall \boldsymbol{x}' \in S^u$ 可知, 如果 $H^{K}(\boldsymbol{x}'^{u})$ 大于当前种群的最小值, 则子区域 $S^u$ 肯定不包含全局最优解, 从而可以可靠排除.

## 2 局部抽象凸区域剖分差分进化算法

抽象凸方法利用一系列分段线性下界支撑面建立目标函数的下界估计松弛模型, 通过枚举松弛模型的极值解得到原目标函数的解. 因此, 本文基于抽象凸分段线性估计特性, 在 DE 算法中, 对部分个体构建分段线性抽象凸下界支撑面, 对搜索区域进行动态剖分, 从而利用区域剖分特性逐步缩小搜索区域, 并通过提取对应区域的下界信息来指导种群进化, 从整体上降低算法计算代价, 提高算法搜索效率和可靠性.

### 2.1　算法设计

如图 1 所示, 通过执行 DE 算法的变异交叉过程生成新个体后, 首先, 判断新个体是否有效 (即是否在有效区域中), 如果新个体为有效个体, 则提取新个体的邻近个体构建分段线性下界支撑面, 对搜索区域进行剖分, 形成下界估计区域及其在目标函数曲面上对应的优化区域, 从而根据区域剖分特性识别出部分无效区域, 同时利用下界估计信息指导种群进化, 并筛选出当前种群中的较差个体. 如果新个体的下界估计值大于目标个体的函数值, 并且下界估计区域的极值大于当前种群的最优值, 则将下界估计区域对应的优化区域视为无效区域, 同时将种群中目标函数值大于新个体下界估计值的个体视为较差个体; 如果新个体的下界估计值小于目标个体的函数值, 且新个体的目标函数值小于目标个体的函数值, 则新个体替换目标个体, 同时根据分段线性下界支撑面的广义下界方向作局部增强. 另外, 根据选择结果对搜索区域再次剖分, 即如果新个体的目标函数值大于目标个体的函数值, 则对新个体构建分段线性下界支撑面, 对原估计区域再次进行剖分, 从而利用各区域的极值与当前种群的最优值进行比较, 进一步识别出无效区域. 最后, 随机选择出部分较差个体, 并根据式 (33) 进行增强处理, 即

$$x^t_{\mathrm{poor},i} = x^t_{\mathrm{poor},i} + F \cdot d^t_i, \quad i = 1, 2, \cdots, N \quad (33)$$

其中, $\boldsymbol{x}^t_{\mathrm{poor}}$ 为较差个体, $\boldsymbol{d}^t$ 为与其对应的局部邻域下降方向, $F$ 为增益常数.

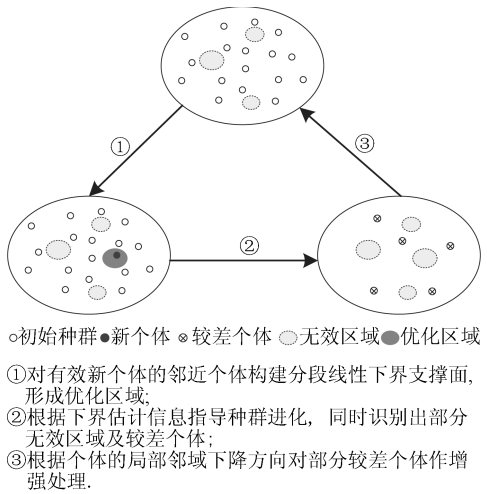



图 1　LPDE 算法基本思想

Fig. 1　The basic idea of LPDE algorithm

通过图 2 所示的 1 维问题来进一步描述算法的区域剖分过程. 假设 A 为目标个体, B 为有效新个体, 则对 B 个体最近的两个个体 C 和 D 构建分段线性下界支撑面, 对可行域进行剖分, 形成 C 和 D 之间的优化区域及其对应的下界估计区域, 从而可以根据下界支撑面计算出个体 B 的下界估计值 $y^{\mathrm{B}}_u$, 因为 $y^{\mathrm{B}}_u$ 大于个体 A 的目标函数值, 则无需对新个体 B 作目标函数评价, 且保留目标个体 A, 同时将 $y^{\mathrm{B}}_u$ 所在的水平线 (面) 定为较差个体评定线 (面), 即位于较差个体评定线 (面) 以上的个体视为较差个体, 如个体 C 和个体 I; 又因为 B 所在的下界估计区域的极小值 $d^{\mathrm{CD}}_u$ 大于当前种群中的最优值, 则根据区域剖分特性可知, 此区域不可能包含全局最优解, 所以可以将个体 C 和 D 之间的区域看作无效区域, 且删除个体 C 和 D 的下界支撑面. 再假设 E 为目标个体, F 为有效新个体, 对其最近的个体 G 和 H 构建分段线性下界支撑面, 以提取 G 与 H 之间的搜索区域, 以及与其对应的下界估计区域, 由于个体 F 的下界估计值 $y^{\mathrm{F}}_u$ 小于个体 E 的目标值, 则需要通过个体 F 的原目标函数值判断个体 F 是否优于个体 E, 由于 F 的目标函数值小于 E 的目标函数值, 则新个体 F 被目标个体 E 取代; 继续计算出个体 F 所在下界估计区域的极值点 $Q(\boldsymbol{x}'^u, d^{\mathrm{GH}}_u)$, 及其在目标函数上对应的点 $Q'(\boldsymbol{x}'^u, g(\boldsymbol{x}'^u))$, 因为 $Q'$ 优于个体 F, 则 $Q'$ 取代个体 F , 同时删除个体 G 和 H 的下界支撑面; 为了进一步缩小搜索区域, 提高算法搜索效率, 所以当个体 F 的目标函数值大于个体 E 的目标函数值时, 对新个体 F 建立分段线性下界支撑面, 将原估计区域剖分成 H 和 F 之间的区域及 F 和 G 之间的区域, 并计算出各区域的极值 $d^{\mathrm{HF}}_u$ 和 $d^{\mathrm{FG}}_u$, 如果对应区域的极值大于当前种群的最小值, 则将此区域视为无效区域, 并删除所有支撑面; 最后, 随机选取部分较差个体, 根据式 (33) 进行增强处理, 即假设较差个体有 $N_j$ 个, 则随机选取 $t$ $(1 \le t \le N_j)$ 个进行增强处理.

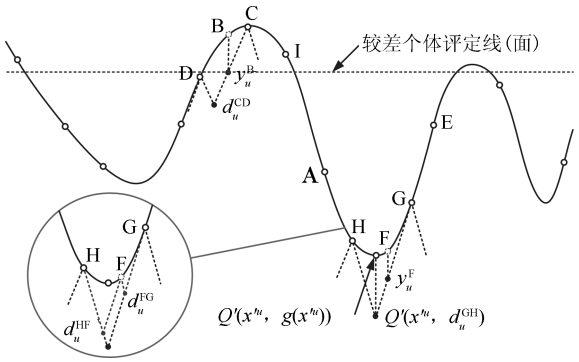



图 2　LPDE 算法区域剖分过程示意图

Fig. 2　The schematic of region partition in LPDE

### 2.2　算法流程

LPDE 算法步骤如下 (最小化问题):

**步骤 1 (初始化).** 设置增益常数 $F$、交叉概率 $C_R$、种群规模 $N_P$ 和常数 $M$ 的值, 随机初始化种

群, 置无效区域为空, 并以单位单纯形区域各顶点的支撑向量组成的初始支撑矩阵为根建立 $n$ 叉树.

**步骤 2 (生成新个体).** 执行 DE 算法的变异和交叉操作生成新个体, 并判断是否为有效新个体, 若为无效新个体则转步骤 11.

**步骤 3 (区域剖分).** 根据式 (14) 对新个体的邻近个体构建分段线性下界支撑面, 从而对搜索区域进行剖分, 同时以各区域对应的支撑矩阵为 $n$ 叉树的节点保存下界估计信息.

**步骤 4 (利用下界信息指导种群更新).** 提取下界估计区域对应的优化区域, 根据式 (13) 计算出新个体的下界估计值, 如果新个体的下界估计值大于目标个体的目标函数值, 则保留目标个体不变, 并继续步骤 5; 否则, 转步骤 7.

**步骤 5 (筛选较差个体).** 根据新个体的下界估计建立较差个体评定线 (面), 找出较差个体, 即如果种群中个体的目标函数值大于新个体的下界估计值, 则将其视为较差个体.

**步骤 6 (识别无效区域).** 先根据式 (27) 计算出新个体所在下界估计区域的极小值, 如果新个体所在区域的极小值大于当前种群的最优值, 则说明此区域不包含全局最优解, 且将此区域对应的优化区域视为无效区域; 然后, 转步骤 10.

**步骤 7 (选择).** 对新个体作目标函数评价, 若新个体优于目标个体, 则继续步骤 8; 否则, 转步骤 9.

**步骤 8 (局部增强).** 根据式 (26) 计算出新个体所在估计区域的极小解, 如果新个体所在估计区域的极小解在目标函数上对应的点优于新个体, 则替换新个体, 并删除所有支撑面, 同时转步骤 11.

**步骤 9 (二次剖分).** 根据选择结果进行二次剖分, 即根据式 (14) 对新个体构建分段线性下界支撑面, 对搜索区域进一步剖分, 通过比较对应区域的极值与当前种群的最优值进一步识别出无效区域, 并删除所有支撑面, 同时转步骤 11.

**步骤 10 (较差个体增强处理).** 先随机选取部分较差个体, 并根据式 (6) 计算出各自的局部邻域下降方向, 然后根据式 (33) 作增强处理.

**步骤 11 (迭代).** 判断是否满足终止条件, 如果满足则保存结果并退出; 否则, 重复步骤 2 ∼ 10.

**算法 1.** 给出了搜索区域剖分过程的伪代码.

本文在步骤 3 中只对新个体邻近的两个个体构建下界支撑面, 并以 $n$ 叉树的形式来保存下界估计值, 而且在更新操作完成以后删除所有支撑面, 实际上树的节点最多只有 4 个, 其中, 找出与新个体最近的两个个体的时间杂度为 $\mathrm{O}(N_P)$, 构建分段线性下界支撑面的时间复杂度为 $\mathrm{O}(N+1)$; 步骤 4 中计算新个体的下界估计信息时, 从 $n$ 叉树中查询包含新个体的叶子节点的时间复杂度为 $\mathrm{O}(\log T)$, 其中, $T$ ($T \le 4$) 为 $n$ 叉树的节点数; 步骤 5 中筛选较差个体的时间复杂度为 $\mathrm{O}(N_P)$; 步骤 6 中找出当前种群的最优个体的时间复杂度为 $\mathrm{O}(N_P)$; 步骤 8 计算新个体所在下界估计区域的极小解的时间复杂度为 $\mathrm{O}(1)$; 步骤 9 中对新个体构建分段线性支撑面的时间复杂度为 $\mathrm{O}(N+1)$; 步骤 11 中对较差个体作增强处理的时间复杂度为 $\mathrm{O}(t)$, 其中, $t$ 为随机选取的较差个体的数目. 算法步骤 2 ∼ 10 中某些步骤在满足条件的情况下才执行, 当步骤 2 ∼ 10 都执行时, 最大时间复杂度为 $\mathrm{O}(N_P)+\mathrm{O}(N+1)+\mathrm{O}(\log T)$, 由于 $\log T \le 0.6$, 则最终时间复杂度为 $\mathrm{O}(N_P)+\mathrm{O}(N+1)$, 且当 $N_P \ge (N+1)$ 时, 时间复杂度为 $\mathrm{O}(N_P)$, 否则为 $\mathrm{O}(N+1)$. 因此, LPDE 算法相对于基本 DE 算法所增加的时间复杂度为 $\mathrm{O}(N_P)$ 或 $\mathrm{O}(N+1)$. 由此可以看出, LPDE 算法并没有明显增加 DE 算法的时间复杂度.

**算法 1.** The procedure of region partition in LPDE

Input: $x_{\mathrm{trial}}$, target individual $x_i$, invalid region $IR$, tree $T$
Output: selection result

```
Begin
    for t ← 1 to m
        Find(x^t_near)//找出与 x_trial 邻近的 m 个个体
        l^t_near = SupportVector (x^t_near)
        UpdateTree(l^t_near)
    end
    Node=FindNode(x_trial) //从树 T 中找出包含 x_trial 的节点
    if (Node ∉ IR)
        evaluate y^trial_u //根据式 (13) 计算 x_trial 的下界估计值
        if (y^trial_u ≥ f(x_i))
            for i ← 1 to N_P
                if(f(x_i) > y^trial_u)
                    x^{N_j}_poor = x_i
                end
            end
            evaluate d_u //根据式 (27) 计算 Node 区域的最小值
            if (d_u ≥ f(x_best))
                IR = IR∪Node
            end
        else
            if (f(x_trial) < f(x_i))
                x_i = x_trial
                evaluate x*_min //根据式 (26) 计算 Node 区域的最小解
                if (f(x*_min) < f(x_trial))
                    x_i = x*_min
                end
            else
                l_trial = SupportVector (x_trial)
            end
        end
    end
    if(N_j ≥ 1)
        for t ← 1 to rand(1, N_j)
            evaluate d^t_poor//根据式 (6) 计算 x^t_poor 的局部邻域下降方向
            enhance(x^t_poor)//根据式 (33) 对 x^t_poor 作增强处理
        end
    end
    delete T
End
```

## 3 数值研究

### 3.1 标准测试函数及参数设置

为了验证所提算法的性能, 本文选用以下 10 个典型标准测试函数进行优化:

1) Sphere 函数

$$f_1(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2$$

2) Tablet 函数

$$f_2(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{N} x_i^2$$

3) Zakharov 函数

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{N} x_i^2 + \left(\sum_{i=1}^{N} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{N} 0.5 i x_i\right)^4$$

4) Rosenbrock 函数

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{N-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$$

5) Griewank 函数

$$f_5(\boldsymbol{x}) = 1 + \frac{1}{4\,000} \sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

6) Schaffer 2 函数

$$f_6(\boldsymbol{x}) = \sum_{i=1}^{N-1} (x_i^2 + x_{i+1}^2)^{0.25} (\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$$

7) Levy and Montalvo 1 函数

$$f_7(\boldsymbol{x}) = \frac{\pi}{N} \left( 10 \sin^2(\pi y_1) + \sum_{i=1}^{N-1} (y_i - 1)^2 \times (1 + 10 \sin^2(\pi y_i + 1)) + (y_N - 1)^2 \right),$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

8) Levy and Montalvo 2 函数

$$f_8(\boldsymbol{x}) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{N-1} (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1}) + (x_N - 1)^2 \times (1 + \sin^2(2\pi x_N))))$$

9) Ackley 函数

$$f_9(\boldsymbol{x}) = -20 \exp\left(-0.02 \sqrt{N^{-1} \sum_{i=1}^{N} x_i^2}\right) - \exp\left(N^{-1} \sum_{i=1}^{N} \cos(2\pi x_i)\right) + 20 + \mathrm{e}$$

10) Rastrigin 函数

$$f_{10}(\boldsymbol{x}) = 10N + \sum_{i=1}^{N} (x_i^2 - 10\cos(2\pi x_i))$$

其中, $f_1 \sim f_4$ 为单模态函数, $f_5 \sim f_{10}$ 为多模态函数, 每个函数问题具有不同的目标函数曲面地形和复杂度, 且多模态函数局部最优解的个数随着维数的增大而增加. 为了保证实验结果的客观性, 10 个测试函数中, 既包含了最优解在原点的函数, 又包含了最优解不在原点的函数, 以防止零点吸引子的影响. 表 1 给出了各测试函数的实验维数 ($N$)、搜索范围、全局最优值及最优解.

表 1　各测试函数的参数

Table 1　The parameters of benchmark founctions

| 函数 | 实验维数 ($N$) | 搜索范围 | 全局最优值 (最优解) |
|---|---|---|---|
| $f_1$ | 10, 30 | $(-100, 100)^N$ | $0(0, \cdots, 0)$ |
| $f_2$ | 10, 30 | $(-100, 100)^N$ | $0(0, \cdots, 0)$ |
| $f_3$ | 5, 10 | $(-5, 10)^N$ | $0(0, \cdots, 0)$ |
| $f_4$ | 2, 4 | $(-30, 30)^N$ | $0(1, \cdots, 1)$ |
| $f_5$ | 10, 30 | $(-600, 600)^N$ | $0(0, \cdots, 0)$ |
| $f_6$ | 10, 30 | $(-100, 100)^N$ | $0(0, \cdots, 0)$ |
| $f_7$ | 10, 30 | $(-10, 10)^N$ | $0(-1, \cdots, -1)$ |
| $f_8$ | 10, 30 | $(-5, 5)^N$ | $0(1, \cdots, 1)$ |
| $f_9$ | 10, 30 | $(-30, 30)^N$ | $0(0, \cdots, 0)$ |
| $f_{10}$ | 5, 10 | $(-5.12, 5.12)^N$ | $0(0, \cdots, 0)$ |

选取 DE、DERL、DELB、CoDE 和 ACUP (Abstract convex underestimate population-based algorithm) 5 种算法与所提算法进行比较实验, 其中, ACUP 算法为基于抽象凸下界估计的群体全局优化算法[26], 算法首先对整个初始群体构建抽象凸下界支撑面, 然后利用不断收紧的下界信息指导种群进化, 最后根据进化信息更新下界支撑面. 实验中, 函数 $f_1 \sim f_3$ 的种群规模为 $N_P = 20$, 函数 $f_5 \sim f_{10}$ 的种群规模为 $N_P = 30$; 鉴于比较的合理性和公平性, 而不考虑其他因素的影响, 将 DE、DERL、DELB、ACUP 及 LPDE 算法中增益常数 $F$ 和交叉概率 $CR$ 均设置为 0.5, 同时在 CoDE 算法的参数池中加入了 $F = 0.5, CR = 0.5$. DELB、CoDE 和 ACUP 算法其他参数设置分别参

见文献 [17, 20, 26]. 此外, 各算法对各测试函数均独立运行 30 次.

### 3.2 评价指标

实验采用如下评价指标:

1) 函数评价次数 (Function evaluations, FES): 当所求得的最优值与函数的全局最优值的误差在规定的范围内时 (本文设置为 0.00001), 算法所进行的函数评价次数.

2) 成功率 (Success rate, SR): 规定所求得的最优值与函数的全局最优值的误差在可接受范围内时为成功, SR = 成功求解次数/总运行次数.

3) 成功表现 (Success performance, SP): SP = 30 次独立运行的平均函数评价次数/成功率.

4) 优化结果性能: 在给定的函数评价次数内 ($1\,000N$), 30 次独立运行所得结果的平均值 (Mean) 和标准差 (Standard deviation, Std); 由于所有测试函数的最优值均为 0, 所以实验中取其对应的对数.

5) 平均收敛速度: 在给定的函数评价次数内, 以函数评价次数为横坐标, 函数值的对数为纵坐标, 30 次独立运行的平均收敛曲线, 由于 0 的对数为负无穷, 所以在纵轴上用 "−Inf" 表示.

### 3.3 实验结果与分析

表 2 给出了各测试函数 30 次独立运行的平均函数评价次数和成功率对比数据, 可以看出, LPDE 算法在对 10 个测试函数 (20 个问题) 进行优化求解时, 除了函数 $f_2$ 和 $f_6$ 外, 其余函数的计算代价 (函数评价次数) 和可靠性 (成功率) 均优于其他 5 种算法, 对于函数 $f_2$ 和 $f_6$, 虽然 LPDE 算法的函数评价次数略高于 DERL 或者 DELB 算法, 但是其成功率均高于 DERL 和 DELB 算法. 另外, 综合 10 个测试函数的平均结果来看 (表 2 最后一行), 由于 LPDE 算法根据区域剖分特性能够安全排除部分无效区域, 并利用下界估计信息指导种群进化, 有效地减少了函数评价次数, 所以其计算代价最小, 平均函数评价次数为 8 787, DE、DERL、DELB、CoDE 和 ACUP 算法分别为 15 186、11 065、11 365、22 195 和 9 741, 也就是说, LPDE 算法相对于 DE、DERL、DELB、CoDE 和 ACUP 算法分别节省了 42.1 %、20.6 %、22.7 %、60.4 % 和 9.8 %; 又由于 LPDE 算法对部分无效区域的排除能够在一定程度上避免算法陷入局部最优, 所以其可靠性最高, 平均成功率为 0.995, CoDE 算法次之, DELB 算法最低.

图 3 为 6 种算法函数评价次数和成功率的箱型对比图. 从图 3 中可以明显看出, CoDE 算法虽然可靠性较高, 仅次于 LPDE 算法, 但是其计算代价最大; DERL 和 DELB 算法相对于 DE 算法来说, 虽然计算代价得到了一定的改善, 但是两者可靠性均较低; 其次, 相对于 ACUP 算法来说, 由于采用局

表 2 函数评价次数和成功率对比数据

Table 2 Compared data on function evaluations and success rates

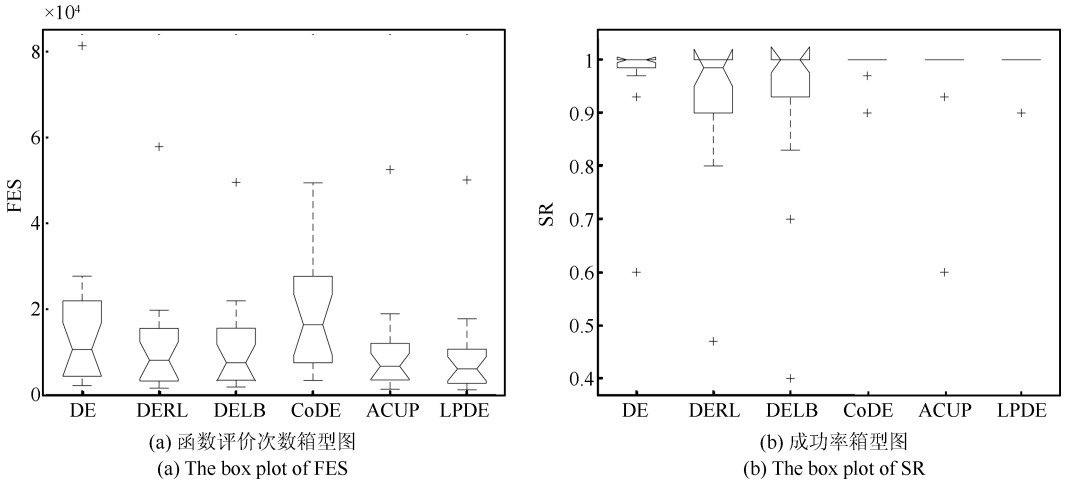
| 函数 | $N$ | DE | | DERL | | DELB | | CoDE | | ACUP | | LPDE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | FES | SR | FES | SR | FES | SR | FES | SR | FES | SR | FES | SR |
| $f_1$ | 30 | 12 094 | 1.00 | 8 765 | 0.90 | 13 747 | 1.00 | 17 174 | 1.00 | 8 938 | 1.00 | 8 432 | 1.00 |
| | 10 | 4 020 | 1.00 | 3 089 | 1.00 | 3 257 | 1.00 | 6 816 | 1.00 | 2 479 | 1.00 | 2 110 | 1.00 |
| $f_2$ | 30 | 12 602 | 1.00 | 9 176 | 0.97 | 14 750 | 1.00 | 17 741 | 1.00 | 11 028 | 1.00 | 9 726 | 1.00 |
| | 10 | 4 491 | 1.00 | 3 389 | 1.00 | 3 601 | 1.00 | 7 449 | 1.00 | 4 054 | 1.00 | 3 513 | 1.00 |
| $f_3$ | 10 | 8 663 | 1.00 | 5 824 | 0.97 | 6 828 | 1.00 | 8 142 | 1.00 | 4 521 | 1.00 | 3 912 | 1.00 |
| | 5 | 2 212 | 1.00 | 1 598 | 1.00 | 1 889 | 1.00 | 3 407 | 1.00 | 1 375 | 1.00 | 1 211 | 1.00 |
| $f_4$ | 4 | 8 339 | 0.60 | 7 475 | 0.47 | 6 971 | 0.90 | 9 897 | 0.97 | 6 817 | 0.60 | 6 521 | 0.90 |
| | 2 | 2 449 | 0.93 | 1 964 | 0.87 | 2 231 | 0.97 | 5 045 | 1.00 | 2 024 | 0.93 | 1 798 | 1.00 |
| $f_5$ | 30 | 22 346 | 0.97 | 15 765 | 0.90 | 13 459 | 0.70 | 28 028 | 1.00 | 13 052 | 1.00 | 11 702 | 1.00 |
| | 10 | 21 551 | 0.97 | 15 369 | 0.90 | 18 550 | 0.97 | 49 423 | 0.90 | 14 094 | 1.00 | 12 109 | 1.00 |
| $f_6$ | 30 | 81 375 | 1.00 | 57 841 | 0.93 | 49 560 | 0.40 | 108 602 | 1.00 | 52 512 | 1.00 | 50 114 | 1.00 |
| | 10 | 22 567 | 1.00 | 17 396 | 1.00 | 18 003 | 1.00 | 44 858 | 1.00 | 18 915 | 1.00 | 17 782 | 1.00 |
| $f_7$ | 30 | 13 294 | 1.00 | 9 510 | 0.97 | 8 787 | 1.00 | 16 713 | 1.00 | 6 519 | 1.00 | 6 002 | 1.00 |
| | 10 | 4 248 | 1.00 | 3 182 | 1.00 | 3 373 | 1.00 | 7 573 | 1.00 | 2 983 | 1.00 | 2 690 | 1.00 |
| $f_8$ | 30 | 13 396 | 1.00 | 9 530 | 1.00 | 8 053 | 0.83 | 16 917 | 1.00 | 7 512 | 1.00 | 7 112 | 1.00 |
| | 10 | 4 121 | 1.00 | 3 093 | 1.00 | 3 191 | 1.00 | 7 263 | 1.00 | 2 386 | 1.00 | 1 509 | 1.00 |
| $f_9$ | 30 | 27 690 | 1.00 | 19 761 | 1.00 | 16 436 | 0.93 | 35 711 | 1.00 | 16 190 | 1.00 | 14 937 | 1.00 |
| | 10 | 9 169 | 1.00 | 6 875 | 1.00 | 6 892 | 1.00 | 16 058 | 1.00 | 6 609 | 1.00 | 6 210 | 1.00 |
| $f_{10}$ | 10 | 23 038 | 1.00 | 17 020 | 0.80 | 21 962 | 0.93 | 27 308 | 1.00 | 8 651 | 1.00 | 5 513 | 1.00 |
| | 5 | 6 062 | 1.00 | 4 681 | 1.00 | 5 754 | 1.00 | 9 766 | 1.00 | 4 162 | 1.00 | 2 835 | 1.00 |
| **AVE** | | 15 186 | 0.974 | 11 065 | 0.934 | 11 365 | 0.932 | 22 195 | 0.994 | 9 741 | 0.977 | **8787** | **0.995** |

部抽象凸区域剖分的 LPDE 算法对无效区域的排除进行了改进, 从而能够找出更多的无效区域, 同时根据个体的局部邻域下降方向对部分较差个体作增强处理, 因此, LPDE 算法不仅计算代价得到了改善, 而且可靠性也得到了提高.

为了进一步验证所提算法在计算代价和可靠性方面的整体优势. 先根据表 2 计算出各测试函数的成功表现 (SP) 值, 再对各测试函数的 SP 值进行归一化处理 (各测试函数所有的 SP 值除以最好算法的 SP 值), 最后绘制成功表现归一化经验分布图[27] 来进行比较分析. 其中, SP 值最小而经验分布值最大的算法为最好算法. 图 4 给出了 10 个测试函数的成功表现归一化经验分布图和 SP 性能表, 可以看出, LPDE 算法在计算代价和可靠性方面的整体性能明显优于其他 5 种算法, ACUP 算法仅次于 LPDE 算法, DELB 算法最差.

表 3 和表 4 分别为单模态函数和多模态函数的优化结果性能对比数据. 表中, 平均值和标准偏差的对数越小越好. 同时, 基于 30 次独立运行的结果, 采用 Wilcoxon signed rank test[28] 非参数假设检验验证所提算法与其他算法之间差异的显著性, 显著性水平为 0.05. 通过上标 “+/ ≈ /−” 分别表示所提算法明显优于所比算法, 所提算法与所比算法没有显著性差异, 所提算法明显差于所比算法.

通过表 3 可以看出, LPDE 算法在对 4 个单模态函数 (8 个问题) 进行求解时, 除了 $f_2$-10 维问题略逊于 DERL 算法外, 其余问题优化结果的性能均优于其他算法, 且 LPDE 算法的标准偏差最小, 由此说明, LPDE 算法在求解单模态函数时比较稳定, 解的质量较高. 此外, 从表 3 最后一行可以看出, 在




(a) 函数评价次数箱型图
(a) The box plot of FES

(b) 成功率箱型图
(b) The box plot of SR

图 3　箱型对比图

Fig. 3　The comparison diagram with box plots

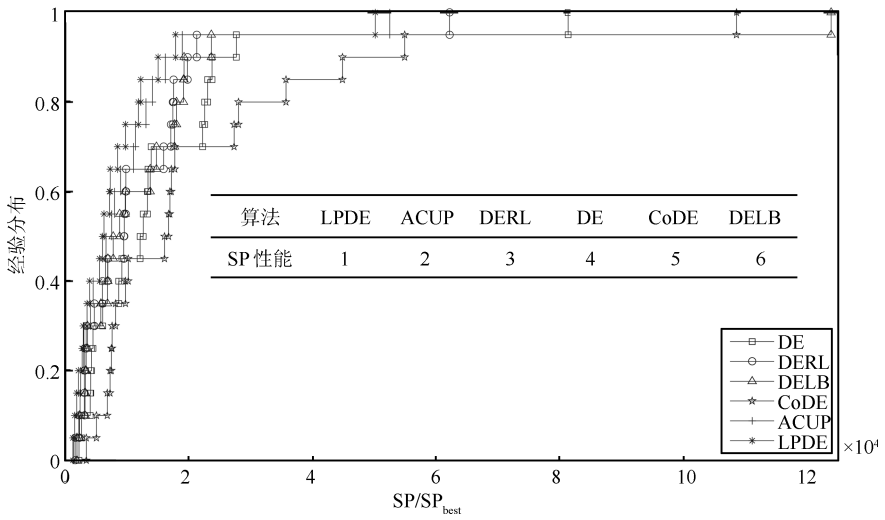



图 4　成功表现归一化经验分布图

Fig. 4　Empirical distribution of normalized success performance

解的显著性方面, LPDE 算法显著优于 DE 和 CoDE 算法所有问题, 且分别显著优于 DERL、DELB 和 ACUP 算法 5 个、7 个和 7 个问题, 除了 $f_2$-10 维问题外, 对于其余问题, 虽然 LPDE 算法与其他算法相比没有显著性差异, 但是其结果均优于其他算法.

图 5 给出了单模态函数的平均收敛速度曲线图, 从图 5 (a) 和图 5 (c) 可以看出, LPDE 算法收敛速度最快, ACUP 算法次之, DERL 算法虽然前期收敛速度仅次于 LPDE 或者 ACUP 算法, 但是由于其可靠性较低, 很快就陷入了局部最优, 出现了早熟收敛; 从图 5 (b) 可以看出, LPDE 算法收敛速度最快, ACUP 算法略逊于之, DE 和 DERL 算法均陷入了局部最优而未能成功求解; 其次, 对于 $f_4$-2 维 Rosenbrock 函数这一经典优化问题, 其函数曲面极其复杂, 算法很难辨别搜索方向, 从而难以找到其全局最优点, 从图 5 (d) 可以看出, 在对 $f_4$-2 维函数进行求解时, DE、DERL、DELB、CoDE 及 ACUP

表 3 单模态函数的优化结果性能对比数据、平均值 (标准差)

Table 3 Compared data of optimization results on unimodal functions, mean (Std)

| 函数 | $N$ | DE | DERL | DELB | CoDE | ACUP | LPDE |
|---|---|---|---|---|---|---|---|
| $f_1$ | 30 | $-10.12(-9.39)^+$ | $-1.29(-0.55)^+$ | $-12.43(-11.81)^+$ | $-11.52(-11.18)^+$ | $-22.24(-21.68)^+$ | $-25.02(-24.45)$ |
| | 10 | $-19.20(-19.01)^+$ | $-26.81(-26.68)^+$ | $-24.14(-23.77)^+$ | $-9.24(-9.26)^+$ | $-28.34(-27.79)^+$ | $-30.54(-30.32)$ |
| $f_2$ | 30 | $-20.09(-20.07)^+$ | $-0.50(-0.24)^\approx$ | $-13.11(-12.48)^+$ | $-11.60(-11.50)^+$ | $-22.95(-22.22)^\approx$ | $-24.68(-23.96)$ |
| | 10 | $-18.63(-18.40)^+$ | $-26.60(-26.09)^-$ | $-24.15(-23.95)^+$ | $-8.25(-8.27)^+$ | $-22.57(-22.21)^+$ | $-25.83(-25.64)$ |
| $f_3$ | 10 | $-6.11(-6.03)^+$ | $-4.85(-4.12)^+$ | $-7.54(-6.96)^+$ | $-5.66(-5.13)^+$ | $-10.91(-10.76)^+$ | $-11.83(-11.80)$ |
| | 5 | $-13.97(-13.65)^+$ | $-20.43(-20.15)^+$ | $-16.97(-16.62)^+$ | $-7.85(-7.83)^+$ | $-22.47(-21.91)^+$ | $-24.08(-23.55)$ |
| $f_4$ | 4 | $-0.40(-0.39)^+$ | $-0.63(-0.16)^\approx$ | $-0.49(-0.05)^\approx$ | $0.08(-0.13)^+$ | $-0.51(-0.59)^+$ | $-0.92(-0.91)$ |
| | 2 | $-1.76(-1.20)^+$ | $-1.96(-1.47)^+$ | $-1.49(-1.08)^+$ | $-0.61(-0.41)^+$ | $-2.94(-2.28)^+$ | $-4.60(-3.98)$ |
| $+/\approx/-$ | | 8/0/0 | 5/2/1 | 7/1/0 | 8/0/0 | 7/1/0 | –/–/– |

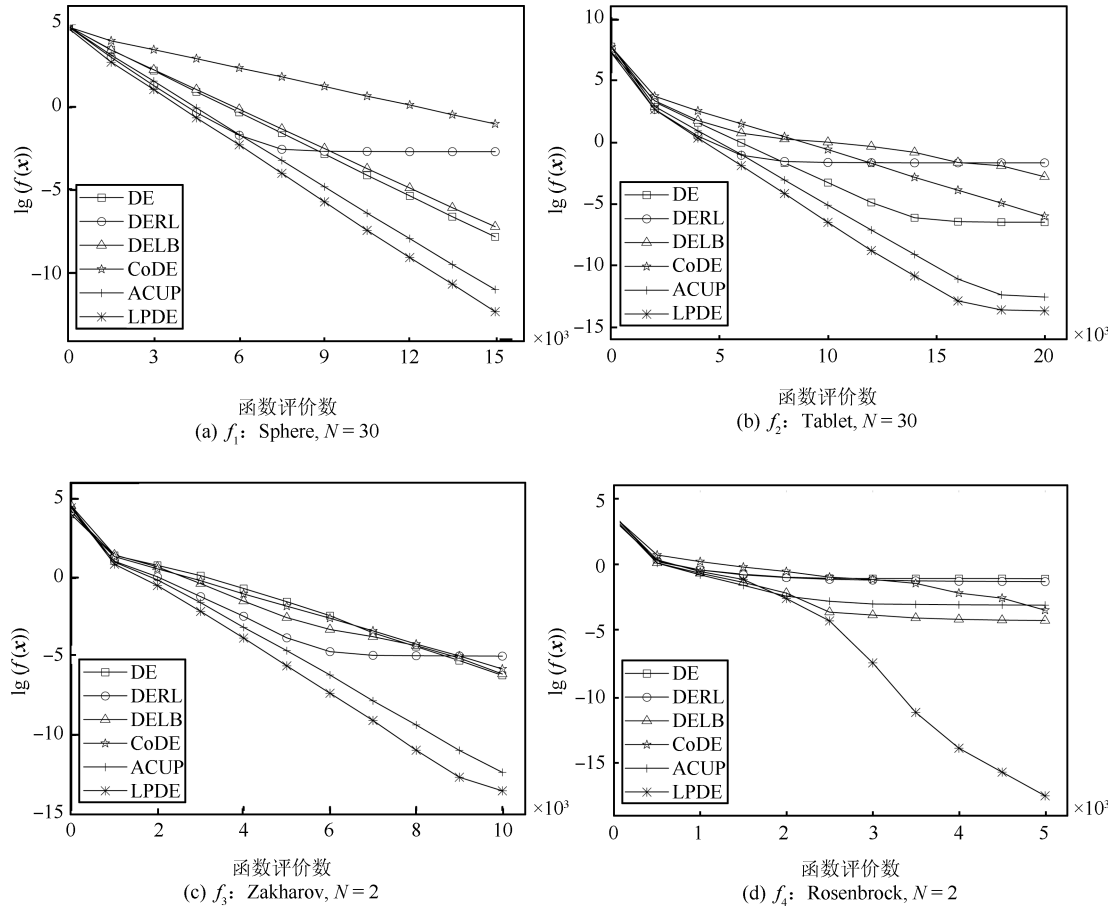



图 5 单模态函数平均收敛速度曲线图

Fig. 5 The curve graph of mean convergence speed on unimodal functions

算法均陷入了局部最优, 出现了早熟收敛而未能成功求解, 只有 LPDE 算法能够非常稳定地向着全局最优点收敛.

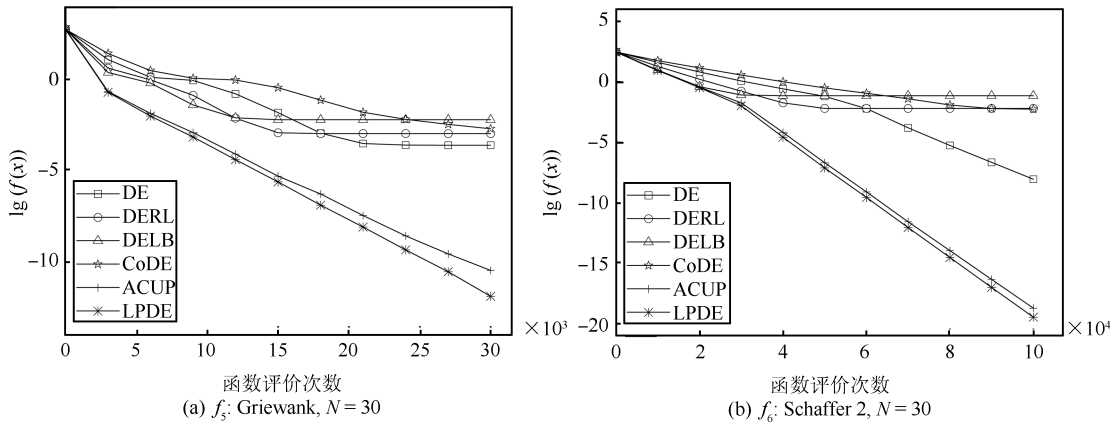
对比表 4 中的数据可以看出, LPDE 算法在对 6 个多模态函数 (12 个问题) 进行优化求解时, 所求得的解的性能均优于其他 5 种算法. 特别是在对函数 $f_5$-30 维和 $f_{10}$ 求解时, LPDE 算法的效果尤其明显, 在优化结果的平均值上优于其他算法很多. 另外, 与单模态函数一样, LPDE 算法的标准偏差也最小, 对于多模态问题, LPDE 算法也比较稳定, 解的质量也较高. 其次, 从表 4 最后一行可以看出, 在解的显著性方面, LPDE 算法显著优于 DE、DERL 和 CoDE 算法所有问题, 并分别显著优于 DELB 和 ACUP 算法 11 个和 10 个问题, 对于其他问题, 虽然 LPDE 算法与 DELB 和 ACUP 算法相比没有显著性差异, 但是其结果均优于 DELB 和 ACUP 算法.

图 6 为多模态函数的平均收敛速度曲线图, 从图 6 (a) 可以看出, LPDE 算法收敛速度最快, 大约在函数评价次数为 25 300 次时就达到了规定的精度 (纵坐标值为 −10), ACUP 算法仅次于 LPDE 算法, DE、DERL、DELB 及 CoDE 算法均由于可靠性较低, 很快就出现了早熟收敛; 图 6 (b) 表明, LPDE 算法收敛速度最快, ACUP 算法仅次于之, DERL、DELB 及 CoDE 算法均陷入了局部最优, DE 算法虽然未陷入局部最优, 但是函数评价次数为 100 000 时, 仍离规定的精度值有一定的距离; 通过图 6 (c) 可以看出, LPDE 算法收敛速度显著优于其他算法, ACUP 算法仅次于之, DERL 算法收敛情况最差, 大约在函数评价次数为 6 000 次时就陷入了局部最优; 从图 6 (d) 和图 6 (e) 可以看出, 仍然是 LPDE 算法收敛速度最快, ACUP 算法仅次于之, 而 DELB 算法虽然前期收敛速度也较快, 但是由于其算法的贪婪性, 很快就出现了早熟收敛而未能成功求解; 对比图 6 (f) 中的曲线可以看出, 虽然 LPDE、ACUP、DERL 和 DELB 算法都能够到达

表 4　多模态函数的优化结果性能对比数据、平均值 (标准差)

Table 4　Compared data of optimization results on multimodal functions, mean (Std)

| 函数 | $N$ | DE | DERL | DELB | CoDE | ACUP | LPDE |
|---|---|---|---|---|---|---|---|
| $f_5$ | 30 | $-7.01(-6.34)^+$ | $-3.31(-2.73)^+$ | $-2.28(-1.82)^+$ | $-2.94(-2.43)^+$ | $-14.84(-14.59)^+$ | $-17.98(-17.59)$ |
| | 10 | $-0.91(-1.13)^+$ | $-1.27(-1.24)^+$ | $-1.03(-1.09)^+$ | $-0.51(-1.19)^+$ | $-2.03(-1.89)^\approx$ | $-2.43(-2.62)$ |
| $f_6$ | 30 | $0.10(-0.60)^+$ | $-0.78(-1.50)^+$ | $-1.11(-0.58)^+$ | $0.63(-0.10)^+$ | $-0.92(-1.61)^+$ | $-0.99(-1.56)$ |
| | 10 | $-0.82(-1.33)^+$ | $-1.55(-2.15)^+$ | $-1.46(-2.02)^+$ | $0.52(-0.15)^+$ | $-1.43(-2.00)^+$ | $-1.69(-2.26)$ |
| $f_7$ | 30 | $-13.65(-13.67)^+$ | $-19.88(-19.85)^+$ | $-19.74(-19.01)^+$ | $-9.84(-10.01)^+$ | $-21.76(-21.24)^\approx$ | $-22.52(-21.94)$ |
| | 10 | $-14.22(-14.09)^+$ | $-19.01(-18.87)^+$ | $-18.73(-18.57)^+$ | $-6.92(-7.03)^+$ | $-19.44(-19.30)^+$ | $-20.85(-20.72)$ |
| $f_8$ | 30 | $-13.77(-13.79)^+$ | $-19.91(-19.95)^+$ | $-3.44(-2.70)^\approx$ | $-3.44(-2.70)^+$ | $-20.69(-20.11)^+$ | $-22.12(-21.82)$ |
| | 10 | $-14.33(-14.44)^+$ | $-18.99(-18.84)^+$ | $-19.15(-19.07)^+$ | $-7.15(-7.33)^+$ | $-20.75(-20.80)^+$ | $-21.06(-21.01)$ |
| $f_9$ | 30 | $-5.98(-6.50)^+$ | $-8.60(-9.41)^+$ | $-1.51(-0.77)^+$ | $-3.88(-4.39)^+$ | $-11.06(-11.43)^+$ | $-12.28(-12.39)$ |
| | 10 | $-6.09(-6.52)^+$ | $-8.34(-8.77)^+$ | $-8.37(-8.71)^+$ | $-2.43(-2.81)^+$ | $-8.93(-9.17)^+$ | $-9.24(-9.31)$ |
| $f_{10}$ | 10 | $0.97(0.42)^+$ | $0.84(0.42)^+$ | $0.99(0.31)^+$ | $1.06(0.43)^+$ | $-4.28(-3.57)^+$ | $-6.42(-5.88)$ |
| | 5 | $-1.63(-1.24)^+$ | $-4.27(-3.74)^+$ | $-2.17(-1.90)^+$ | $0.18(-0.10)^+$ | $-12.91(-12.56)^+$ | $-14.35(-13.99)$ |
| $+/\approx/-$ | | 12/0/0 | 12/0/0 | 11/1/0 | 12/0/0 | 10/2/0 | −/−/− |




(a) $f_5$: Griewank, $N = 30$　　(b) $f_6$: Schaffer 2, $N = 30$

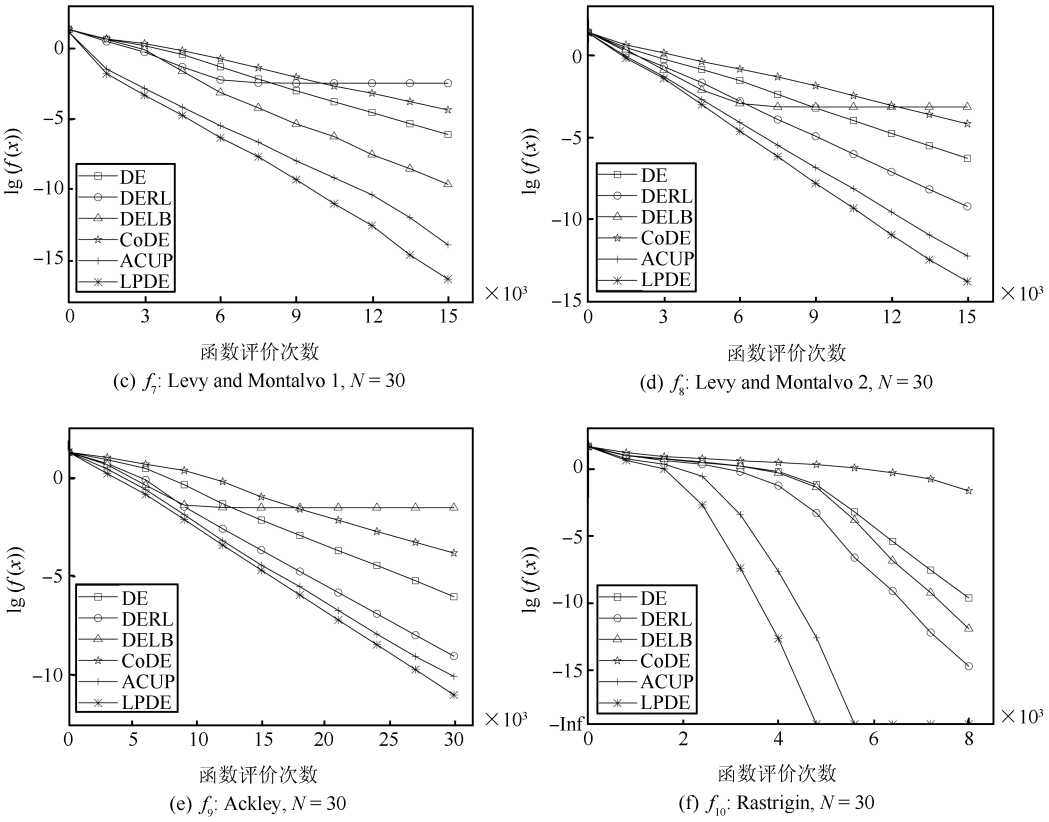



图 6 多模态函数平均收敛速度曲线图

Fig. 6 The curve graph of mean convergence speed on multimodal functions

规定的精度, 但是 LPDE 算法收敛速度明显优于其他算法, 大约在函数评价次数为 4 800 次时就已到达了函数的全局最优值 0, CoDE 算法收敛速度最慢, 函数评价次数为 8 000 次时, 求得的最优值仍离所规定的精度值甚远.

总体来说, 采用局部抽象凸区域剖分的 LPDE 算法, 无论对于单模态函数问题还是多模态函数问题, 所表现出的整体性能优于其他 5 种算法, 不仅计算代价低, 收敛速度快, 而且不易陷入局部最优, 可靠性较高.

## 4 结论

基于抽象凸分段线性估计特性, 本文提出了一种局部抽象凸区域剖分差分进化算法 (LPDE). 所提算法通过构建目标函数的局部分段线性下界估计支撑面, 对可行域进行动态剖分, 从而不仅可以利用区域剖分特性有效地识别出无效区域, 并筛选出部分较差个体, 而且可以合理地利用下界估计信息指导种群进化. 通过 10 个典型的标准测试函数验证了 LPDE 算法计算代价、可靠性及收敛速度方面的优越性能. 另外, 该方法也可以应用到蚁群算法、遗传算法及萤火虫算法等群体算法中. 下一步的主要工作将集中在 LPDE 算法在复杂实际优化问题中的应用研究.

## References


1 Chen Bao-Lin. *Theory and Methods of Optimization* (2nd Edition). Beijing: Tsinghua University Press, 2005.
(陈宝林. 最优化理论与算法. 第 2 版. 北京: 清华大学出版社, 2005.)

2 Walsh G R. *Methods of Optimization*. London: Wiley Press, 1975.

3 Nelder J A, Mead R. A simplex method for function minimization. *The Computer Journal*, 1965, **7**(4): 308−313

4 Adjiman C S, Dallwig S, Floudas C A, Neumaier A. A global optimization method, $\alpha$BB, for general twice-differentiable constrained NLPs: I. Theoretical advances. *Computers & Chemistry Engineering*, 1998, **22**(9): 1137−1158

5 Adjiman C S, Androulakis I P, Floudas C A. A global optimization method, $\alpha$BB, for general twice-differentiable constrained NLPs: II. Implementation and computational results. *Computers & Chemistry Engineering*, 1998, **22**(9): 1159−1179

6 Skjäl A, Westerlund T, Misener R, Floudas C A. A generalization of the classical $\alpha$BB convex underestimation via diagonal and nondiagonal quadratic terms. *Journal of Optimization Theory and Applications*, 2012, **154**(2): 462−490

7 Beliakov G. Cutting angle method — a tool for constrained global optimization. *Optimization Methods and Software*, 2004, **19**(2): 137−151

8 Bagirov A M, Rubinov A M. Cutting angle method and a local search. *Journal of Global Optimization*, 2003, **27**(2−3): 193−213

9 Beliakov G. Geometry and combinatorics of the cutting angle method. *Optimization*, 2003, **52**(4−5): 379−394

10 Floudas C A, Gounaris C E. A review of recent advances in global optimization. *Journal of Global Optimization*, 2009, **45**(1): 3−38

11 Das S, Suganthan P N. Differential evolution: a survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 2011, **15**(1): 4−31

12 Wang Da-Zhi, Liu Shi-Xin, Guo Xi-Wang. A multi-agent evolutionary algorithm for solving total tardiness permutation flow-shop scheduling problem. *Acta Automatica Sinica*, 2014, **40**(3): 548−555
(王大志, 刘士新, 郭希旺. 求解总拖期时间最小化流水车间调度问题的多智能体进化算法. 自动化学报, 2014, **40**(3): 548−555

13 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

14 Islam S M, Das S, Ghosh S, Roy S, Suganthan P N. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 2012, **42**(2): 482−500

15 Hu Rong, Qian Bin. A hybrid differential evolution algorithm for stochastic flow shop scheduling with limited buffers. *Acta Automatica Sinica*, 2009, **35**(12): 1580−1586
(胡蓉, 钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法. 自动化学报, 2009, **35**(12): 1580−1586)

16 Stoean C, Preuss M, Stoean R, Dumitrescu D. Multimodal optimization by means of a topological species conservation algorithm. *IEEE Transactions on Evolutionary Computation*, 2010, **14**(6): 842−864

17 Kaelo P, Ali M M. A numerical study of some modified differential evolution algorithm. *European Journal of Operational Research*, 2006, **169**(3): 1176−1184

18 Cai Y Q, Wang J H. Differential evolution with neighborhood and direction information for numerical optimization. *IEEE Transactions on Cybernetics*, 2013, **43**(6): 2202−2215

19 Bhattacharya A, Chattopadhyay P K. Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch. *IEEE Transactions on Power Systems*, 2010, **25**(4): 1955−1964

20 Wang Y, Cai Z X, Zhang Q F. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 2011, **15**(1): 55−66

21 Gong W Y, Cai Z H. Differential evolution with ranking-based mutation operators. *IEEE Transactions on Cybernetics*, 2013, **43**(6): 2066−2081

22 Zhang Gui-Jun, He Yang-Jun, Guo Hai-Feng, Feng Yuan-Jing, Xu Jian-Ming. Differential evolution algorithm for multimodal optimization based on abstract convex underestimation. *Journal of Software*, 2013, **24**(6): 1177−1195
(张贵军, 何洋军, 郭海锋, 冯远静, 徐建明. 基于广义凸下界估计的多模态差分进化算法. 软件学报, 2013, **24**(6): 1177−1195)

23 Deng Yong-Yue, Zhang Gui-Jun. Multimodal optimization based on local abstract convexity support hyperplanes. *Control Theory & Applications*, 2014, **31**(4): 458−466
(邓勇跃, 张贵军. 基于局部抽象凸支撑面的多模态优化算法. 控制理论与应用, 2014, **31**(4): 458−466)

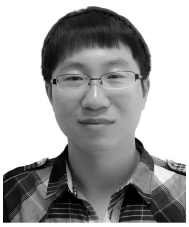
24 Rubinov A M. Abstract convexity and global optimization. *Nonconvex Optimization and Its Applications*. Dordrecht: Kluwer Academic Publishers, 2000.

25 Bagirov A M, Rubinov A M. Global minimization of increasing positively homogeneous functions over the unit simplex. *Annals of Operations Research*, 2000, **98**(1−4): 171−187

26 Zhang Gui-Jun, Zhou Xiao-Gen. Population-based global optimization algorithm using abstract convex underestimate. *Control and Decision*, 2015, **30**(6): 1116−1120
(张贵军, 周晓根. 基于抽象凸下界估计的群体全局优化算法. 控制与决策, 2015, **30**(6): 1116−1120)

27 Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 2009, **13**(2): 398−417

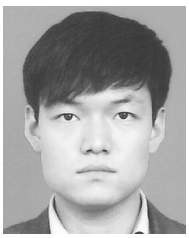
28 Corder G W, Foreman D I. *Nonparametric Statistics for Non-statisticians: a Step-by-step Approach*. Hoboken, NJ: Wiley Press, 2009.


**周晓根** 浙江工业大学信息工程学院博士研究生. 主要研究方向为智能信息处理, 优化理论及算法设计.
E-mail: zhouxiaogen53@126.com
(**ZHOU Xiao-Gen** Ph. D. candidate at the College of Information Engineering, Zhejiang University of Technology. His research interest covers intelligent information processing, optimization theory, and algorithm design.)



**张贵军** 浙江工业大学信息工程学院教授. 主要研究方向为智能信息处理, 优化理论及算法设计, 生物信息学. 本文通信作者. E-mail: zgj@zjut.edu.cn
(**ZHANG Gui-Jun** Professor at the College of Information Engineering, Zhejiang University of Technology. His research interest covers intelligent information processing, optimization theory and algorithm design, and bioinformatics. Corresponding author of this paper.)


**郝小虎** 浙江工业大学信息工程学院硕士研究生. 主要研究方向为智能优化, 生物信息学. E-mail: HXH_zjut@163.com
(**HAO Xiao-Hu** Master student at the College of Information Engineering, Zhejiang University of Technology. His research interest covers intelligent optimization and bioinformatics.)