第 32 卷第 3 期
2015 年 3 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 32 No. 3
Mar. 2015

DOI: 10.7641/CTA.2015.40525

基于抽象凸估计选择策略的差分进化算法

周晓根, 张贵军†, 梅　珊, 明　洁

(浙江工业大学 信息工程学院, 浙江 杭州 310023)

摘要: 针对传统差分进化算法计算代价、可靠性及收敛速度的问题, 提出一种基于抽象凸估计选择策略的差分进化算法(DEUS). 首先, 通过提取新个体的邻近个体建立局部抽象凸下界松弛模型; 然后, 利用下界松弛模型估计目标函数值来指导种群更新, 同时利用下界估计区域极值点快速枚举算法系统排除部分无效区域; 最后, 借助线性拟凸包络的广义下降方向有效地实现局部增强. 12个标准测试函数的实验结果表明, 所提算法计算代价、可靠性及收敛速度均优于DE及DERL, DELB, SaDE等改进算法.

关键词: 差分进化; 全局优化;下界估计; 抽象凸; 支撑向量

中图分类号: TP391　　**文献标识码**: A

Differential evolution algorithm based on abstract convex underestimate selection strategy

ZHOU Xiao-gen, ZHANG Gui-jun†, MEI Shan, MING Jie

(College of Information Engineering, Zhejiang University of Technology, Hangzhou Zhejiang 310023, China)

Abstract: To solve the problems of computational cost, success rate and convergence speed in the conventional differential evolution algorithm, we propose a new differential evolution algorithm based on abstract convex underestimate selection strategy (DEUS). Firstly, the local abstract convex lower relaxed model is constructed by extracting the neighboring individuals of the new individual. Then, the underestimate values which are estimated through the lower relaxed model are used to guide the update process of the population, and some invalid regions of the domain where the global optimum cannot be found are systematically excluded by using the fast enumeration algorithm of the local minimum in the underestimate regions. Finally, the generalized descent directions of the linear quasi convex envelope are employed for local enhancement. Experiments results of 12 benchmark functions show that the proposed algorithm is superior to DE, DERL, DELB and SaDE algorithm in terms of computational cost, success rate and convergence speed.

Key words: differential evolution; global optimization; underestimate; abstract convex; support vector

1　引言(Introduction)

全局优化作为最优化学科领域中一个独立的学科分支, 已成为人们研究实际问题时进行建模和分析的重要手段之一. 在科学、经济和工程中, 如机械设计、生物信息学、环境工程、化学工程设计和控制以及图形处理等, 许多进展都依赖于计算相应优化问题的全局最优解的数值技术. 同时, 随着工程优化问题的规模增大, 优化问题的目标函数的性态也变得越来越复杂, 通常是不连续、不可微、高度非线性的, 没有明确的解析表达式, 且具有多个峰值、多目标的特征. 因此, 解决复杂的优化问题已成为当前计算机科学和优化领域的一个挑战性课题[1].

进化算法是基于自然选择和遗传等生物进化机制的一种随机搜索算法, 已被成功用于求解各种优化问题. 典型的进化算法包括差分进化算法(DE)[2]、遗传算法(GA)[3]、进化策略(ES)[4]、进化规划(EP)[5]以及粒子群算法(PSO)[6]等, 这些算法不需要导数信息, 对函数的性态没有要求, 而且适用范围广、鲁棒性强. Storn和Price[2]提出的DE算法, 已经被证明是进化算法中简单而最高效的随机性全局优化算法, DE算法通过群体内个体间的合作与竞争产生的群体智能指导优化搜索, 具有算法通用, 不依赖于问题信息, 原理简单, 易于实现, 记忆个体最优解和种群内信息共享以及较强的全局收敛能力等特点, 虽然在通信、电力系

收稿日期: 2014−06−07; 录用日期: 2014−12−24.

†通信作者. E-mail: zgj@zjut.edu.cn; Tel.: +86 13958125042.

国家自然科学基金项目(61075062, 61379020), 浙江省自然科学基金项目(LY13F030008), 浙江省科技厅公益项目(2014C33088), 浙江省重中之重学科开放基金资助项目(20120811), 杭州市产学研合作项目(20131631E31)资助.

Supported by National Natural Science Foundation of China (61075062, 61379020), 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 Cooperation Project of Industry-Academia-Research Institute of Hangzhou (20131631E31).

统、化工、光学、生物信息学、模式识别及机械工程等领域的广泛应用中展现出了其独特的优势[7–8], 但是也暴露出诸多不足和缺陷, 如计算代价(如函数评价次数)较高, 收敛速度较慢, 极易陷于局部最优解, 可靠性较低[9].

为了提高DE算法的性能, 国内外学者相继提出了一些改进算法. 针对DE算法计算代价较高的问题, Ali等[10]提出基于锦标赛机制的改进算法(DERL)和采用反射与收缩算子的改进算法(DELB), 以改善计算代价问题; Qin等[11]利用均匀分布对变异率和交叉率进行了调整, 提出一种自适应差分进化算法(SaDE), 通过一种学习过程来自适应调整变异策略及其参数, 从而使得进化过程的不同阶段自适应的调整变异策略及参数, 以降低算法的计算代价; Gao等[12]提出一种基于混沌系统的混合差分进化算法(CS–DE1), 首先通过一种混沌系统初始化种群, 使得个体尽量分散, 然后对DE算法迭代一定的次数, 找出当前种群中的最优个体, 再从此最优个体出发执行Nelder-Mead单纯形搜索, 将种群中一半的较差个体重新初始化, 从而降低算法的计算代价.

针对DE算法收敛速度较慢的问题, Chiou等[13]提出一种基于蚁群方向的混合差分进化算法(ADHDE), 利用蚁群搜索算法实时地从多种变异算子中为DE选择合适的变异操作算子, 从而加快算法的收敛速度; Bhattacharya等[14]将差分进化算法与生物地理学优化算法结合(DE/BBO), 利用BBO算法的迁移和变异操作来提高DE算法的搜索能力, 以加快算法的收敛速度; Elsayed等[15]提出一种新的改进自适应差分进化算法(ISAMODE–CMA), 将种群分成很多亚种群, 对每个亚种群设置相应的进化策略和交叉参数, 并引入一种学习策略来自适应调整每个亚种群的种群规模, 同时利用协方差适应矩阵来加快算法的收敛速度.

针对DE算法极易陷入局部最优解, 出现早熟收敛的问题, Wang等[16]引入一种加速和迁移操作, 加速操作利用梯度信息将最优个体引向更优的区域, 当种群的分散度低于一定的阈值时, 利用迁移操作在最优个体附近区域重新产生新个体, 并替换旧个体, 从而维持种群的多样性, 防止算法早熟收敛; Song等[17]提出一种基于差分进化和分布式估计的混合优化算法(DE–EDA), 利用EDA和DE算法混合来生成后代群体, 同时利用一种改进变异策略来提高种群的多样性, 以防止算法早熟收敛; Rahnamayan等[18]提出一种反向差分进化算法(ODE), 引入一种反向学习技术来初始化种群并产生跳变, 从一定程度上防止算法早熟收敛.

上述改进算法通过改进变异和交叉过程或引入新的操作取得了一定效果, 但是对于一些大规模的高维优化问题, 计算代价及收敛速度仍然是算法的瓶颈所在, 而且也极容易陷于局部最优解. 相关文献研究表明[19], 在DE算法选择过程中, 一定的选择压力能够保证算法的收敛速度, 过大反而会导致早熟收敛而降低可靠性, 过小又会使得收敛速度变慢, 两者相互矛盾. 因此, 如何在加快算法收敛速度的同时提高可靠性是一个关键问题. 本文在DE算法的选择环节引入抽象凸估计策略, 提出一种基于抽象凸估计选择策略的差分进化算法(DEUS). 通过提取新个体的邻近个体建立局部抽象凸下界松弛模型, 进而获取下界估计信息来指导种群更新, 有效减少目标函数评价次数; 同时通过快速枚举下界估计区域的极值信息系统排除部分无效区域, 从而提高算法的可靠性, 减小计算代价, 加快收敛速度; 此外, 借助线性拟凸包络的广义下降方向作局部增强, 进一步加快算法的收敛速度.

2　研究动机(Research motivation)

抽象凸理论[20–22]作为确定性全局优化的一个重要部分, 泛化了凸分析中仿射弱函数的概念. 抽象凸方法则基于一类非凸函数都是它一系列简单弱函数的上包络这一重要结论, 并引入次梯度这一有力分析工具, 利用一系列简单弱函数建立原目标问题的松弛模型, 从而可以通过松弛模型来估计原目标函数.

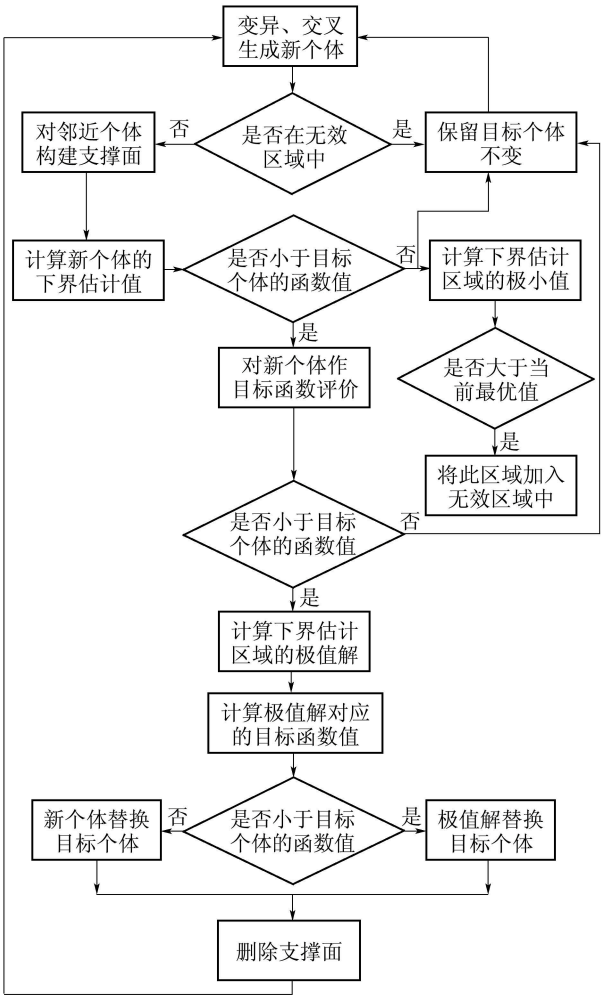

图 1　DEUS算法选择操作流程图

Fig. 1　The flow diagram of selection operation in DEUS

本文在基于局部抽象凸支撑面的多模态优化算法等[23–24]前期研究的基础上, 进一步提出一种基于抽象凸估计选择策略的差分进化算法. 如图1所示, 算法通过模型转换将原目标函数转化为单位单纯形约束条件下的目标函数后, 在DE算法的选择环节, 基于抽象凸理论, 通过提取有效区域中新个体的邻近个体构建抽象凸下界支撑面建立目标函数的局部下界松弛模型, 从而可以通过松弛模型估计新个体的下界信息, 进而根据下界估计信息有选择性的对新个体作目标函数评价, 即利用新个体的下界信息与目标个体的函数值比较来判断是否有必要对新个体作目标函数评价, 而无需对每个新个体都作函数评价, 有效地减少了函数评价次数; 同时, 通过高效模型求解算法获取下界估计区域的极值信息, 进而根据下界估计区域的极值信息系统排除无效区域(即在该区域不包含全局最优解), 避免了对无效区域中个体的评价, 不仅进一步减小了算法的计算代价, 还加快了算法的收敛速度, 而且在一定程度上避免了早熟收敛现象; 其次, 利用线性拟凸包络的广义下降方向作局部增强, 进一步加快了算法的收敛速度. 由于算法只对新个体的邻近个体构建抽象凸下界支撑面, 且更新环节结束后立即删除, 所以算法并没有产生较高的复杂度.

3 抽象凸估计选择策略差分进化算法(DEUS algorithm)

3.1 松弛模型(Relaxed model)

所提算法通过支撑函数组成的松弛模型来估计原目标问题, 进而利用目标问题的下界信息来指导种群更新, 并根据下界估计区域的极值信息系统排除部分无效区域, 同时借助线性拟凸包络的广义下降方向做局部增强, 因此, 松弛模型的建立至关重要.

3.1.1 模型转换(Model transformation)

定义 1 如果函数$f: x \to \mathbb{R}$对于任意的$x_1, x_2 \in D$都满足

$$|f(x_1) - f(x_2)| \leqslant C||x_1 - x_2||, \tag{1}$$

则称函数f在可行域空间D内Lipschitz连续, 其中C称为Lipschitz常数.

定义 2 如果函数f满足以下两个条件:

$$\forall x, y \in \mathbb{R}_+^N,\ x \geqslant y \Rightarrow f(x) \geqslant f(y), \tag{2}$$

$$\forall x \in \mathbb{R}_+^N,\ \lambda \in \mathbb{R}_{++}: f(\lambda x) = \lambda f(x), \tag{3}$$

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

设全局优化问题

$$\min f(x),\ x \in D \subset \mathbb{R}^N, \tag{4}$$

满足定义1, $D = \{x_i | a_i \leqslant x_i \leqslant b_i, i = 1, \cdots, N\}$为函数$f$的可行域空间, $x = (x_1, x_2, \cdots, x_N)^{\mathrm{T}}$为可行域空间$D$内的$N$维连续优化变量, 函数$f$为定义在可行域空间$D$上的目标函数, 在给定的可行域空间内可能包含多个全局最优解和一系列局部最优解.

对原优化变量$x = (x_1, x_2, \cdots, x_N)^{\mathrm{T}}$作如下线性变换:

$$\begin{cases} x_i' \equiv \dfrac{x_i - a_i}{\sum\limits_{i=1}^{N}(b_i - a_i)}, \\ x_{N+1}' \equiv 1 - \sum\limits_{i=1}^{N} x_i', \end{cases} \tag{5}$$

其中: $i = 1, 2, \cdots, N$, $x' = (x_1', x_2', \cdots, x_{N+1}')^{\mathrm{T}}$为定义在单位单纯形区域空间$S \equiv \{x' \in \mathbb{R}_+^{N+1}, x_i' \geqslant 0, \sum\limits_{i=1}^{N+1} x_i' = 1\}$中的线性转换变量.

对式(5)进行反变换有

$$x_i = x_i' \sum_{i=1}^{N}(b_i - a_i) + a_i,\ i = 1, 2, \cdots, N. \tag{6}$$

将式(6)代入式(4), 可得

$$\min \bar{f}(x'),\ x' \in S \subset \mathbb{R}_+^{N+1}, \tag{7}$$

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.

3.1.2 模型建立(Model construction)

文献[20]表明, 任一Lipschitz函数都可以通过加上常数$M(M \geqslant 2C)$转化为IPH函数, 且不改变函数的最优值. 假设式(7)目标函数$\bar{f}$已通过加上常数M转化为IPH函数, 即$\bar{g} = \bar{f} + M$. 假定有K个支撑函数, 基于给定的点$(x'^k, \bar{g}(x'^k))$, $k = 1, 2, \cdots, K$, 令$I = \{1, 2, \cdots, N+1\}$, 则目标函数$\bar{g}$的松弛模型为

$$H^K(x') = \max_{k \leqslant K} \min_{i \in I} l_i^k x_i', \tag{8}$$

第k个支撑函数为

$$h^k(x') = \min_{i \in I} l_i^k x_i' = \bar{g}(x'^k) \min\{\frac{x_1'}{x_1'^k}, \cdots, \frac{x_{N+1}'}{x_{N+1}'^k}\}, \tag{9}$$

其中

$$l^k = (\frac{\bar{g}(x'^k)}{x_1'^k}, \frac{\bar{g}(x'^k)}{x_2'^k}, \cdots, \frac{\bar{g}(x'^k)}{x_{N+1}'^k}) \tag{10}$$

称为支撑向量.

引理 1 根据定义1, 设$\exists C > 0$, 使得式(7)的目标函数$\bar{f}: S \to \mathbb{R}$满足

$$C = \inf_{x_1' \neq x_2'} \frac{|\bar{f}(x_1') - f(x_2')|}{||x_1' - x_2'||_1},\ \forall x_1', x_2' \in S,$$

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

$$h^y(x') = \min_{i \in \Theta(l)} l_i x_i',\ \forall x' \in S,$$

满足$h^y(x') \leqslant \bar{f}(x') + M = \bar{g}(x')$, $\forall x' \in S$. 其中: $\|x'_1 - x'_2\|_1 \equiv \max\limits_{i\in I} \|x'_1 - x'_2\|$, $l = \bar{g}(x')/x'$, $l=(l_1, l_2, \cdots, l_{N+1}) \in \mathbb{R}_+^{N+1}$, 索引集合$\Theta(l) = \{i : l_i > 0\}$.

证　详细证明参见文献[24].

推论 1　设$y^1, y^2, \cdots, y^K \in S$为式(7)目标函数的已知采样点, 则松弛模型

$$H^K(x') = \max_{k=1,\cdots,K} h^k(x') = \max_{k=1,\cdots,K} \min_{i\in\Theta(l)} l_i^k x'_i \tag{11}$$

满足

$$\bar{g}(x') = \bar{f}(x') + M \geqslant H^K(x'),\ \forall x' \in S,$$
$$\bar{g}(x') = \bar{f}(x') + M = H^K(x'),\ \forall x' \in \{y^1, \cdots, y^K\},$$

其中: $l_i^k = (\bar{f}(y^k)+M)/(y_i^k)$, $\Theta(l) = \{i : l_i > 0\}$, 记$h^k(x') \equiv h^{y^k}(x')$, $k = 1, 2, \cdots, K$.

证　根据引理1可知

$$\bar{f}(x') + M \geqslant h^y(x'),\ y \in S,\ \forall x' \in S,$$

故可得到

$$\bar{f}(x') + M \geqslant \max_{k=1,\cdots,K} h^k(x') = H^K(x'),\ \forall x' \in S.$$

设$x' = y^\xi$, $\xi \in 1, 2, \cdots, K$, 则$\bar{f}(y^\xi) + M = h^\xi(y^\xi)$. 另外, 由式(11)可知

$$h^\xi(y^\xi) = H^k(y^\xi),$$

故$\bar{f}(x') + M = H^K(x')$, $\forall x' \in \{y^1, \cdots, y^K\}$.

证毕.

3.1.3　模型求解(Model solution)

根据模型转换式(5)和引理1将原目标函数转化为单位单纯形约束下的IPH函数后, 从而可以利用高效的模型求解算法进行求解. 考虑一个含有K个支撑向量的集合$\Lambda^K = \{l^k\}_{k=1}^K$, 则满足式(12)−(13)的支撑向量集$L = \{l^{k_1}, l^{k_2}, \cdots, l^{k_{N+1}}\}$称为一个有效的支撑矩阵.

$$\forall i, j \in I,\ i \neq j : l_i^{k_i} < l_i^{k_j}, \tag{12}$$

$$\forall v \in \Lambda^K \backslash L,\ \exists i \in I : l_i^{k_i} \geqslant v_i. \tag{13}$$

假设一个$N+1$维有效支撑矩阵为

$$L = \begin{pmatrix} l_1^{k_1} & l_2^{k_1} & \cdots & l_{N+1}^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \cdots & l_{N+1}^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_{N+1}} & l_2^{k_{N+1}} & \cdots & l_{N+1}^{k_{N+1}} \end{pmatrix}, \tag{14}$$

则该支撑矩阵对应的估计区域的极值解$x'_{\min}$和极值d分别为[22]

$$x'_{\min}(L) = \frac{\mathrm{diag}\{L\}}{\mathrm{tr}(L)}, \tag{15}$$

$$d(L) = H^K(x'_{\min}) = \frac{1}{\mathrm{tr}(L)}. \tag{16}$$

3.2　抽象凸估计选择策略(Selection strategy with abstract convex underestimate)

经过变异交叉生成新个体后, 首先判断新个体是否在无效区域中, 如果在, 则无需对新个体作目标函数评价, 且保留目标个体不变, 否则对新个体的邻近个体构建支撑向量, 并且设计n叉树保存各下界估计值, 进而利用新个体的下界信息与目标个体的函数值进行比较来决定是否有必要对新个体作目标函数评价; 如果新个体的下界信息大于目标个体的函数值, 则无需对新个体作目标函数评价, 且保留目标个体不变, 并继续计算出新个体所在的下界估计区域的极小值, 如果下界估计区域的极小值大于当前种群的最小值, 则将此区域视为无效区域; 如果新个体的下界信息小于目标个体的函数值, 则对新个体作目标函数评价, 若新个体优于目标个体, 则新个体替代目标个体, 并利用线性拟凸包络的广义下界方向作局部增强.

以图2所示的1维Shubert问题为例, 假设A为目标个体, B为有效区域中的新个体, 找出离B个体最近的两个个体C和D, 并构建下界支撑面, 计算出B个体的下界估计值$y_{\mathrm{u}}^{\mathrm{B}}$, 因为$y_{\mathrm{u}}^{\mathrm{B}}$大于A个体的目标函数值, 则无需对新个体B作目标函数评价, 且保留目标个体A; 继续计算出B个体所在下界估计区域的极小值d_{u}, 如果d_{u}大于当前种群的最小值, 则将B个体所在的区域(即C与D之间的区域, 不包括C和D)看作无效区域, 并记录下来, 且删除C和D个体的下界支撑面. 再假设E为目标个体, F为新个体且不在无效区域中, 针对离其最近的两个个体G和H构建下界支撑面, 并计算出F个体的下界估计值$y_{\mathrm{u}}^{\mathrm{F}}$, 因为$y_{\mathrm{u}}^{\mathrm{F}}$小于E个体的目标函数值, 则对新A个体F作目标函数评价, 由于F个体优于E个体, 则F个体取代目标个体E. 为了进一步加快算法的收敛速度, 继续计算出F个体所在区域的下界支撑函数的极值点$Q(x_{\mathrm{u}}, d(x_{\mathrm{u}}))$, 及其在目标函数上对应的点$Q'(x_{\mathrm{u}}, f(x_{\mathrm{u}}))$, 因为$Q'$对应的目标函数值小于F个体的目标函数值, 则$Q'$取代F个体, 同时删除G和H个体的下界支撑面.

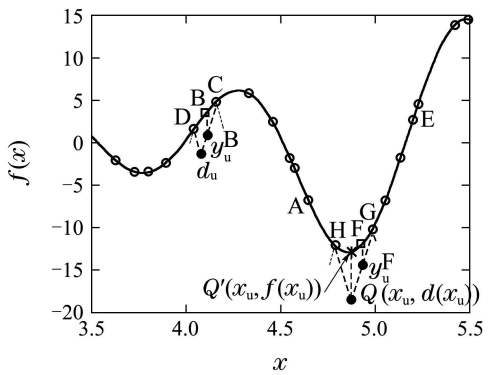

图 2　DEUS算法选择过程示意图

Fig. 2　The diagram of selection procedure in DEUS

3.3 算法描述(Algorithm description)

以最小化问题为例, 假设$f(x)$为N维目标函数, 结合图1, 整体算法描述如下:

Step 1 初始化: 设置常数M, 增益常数F, 交叉概率$C_{\rm R}$, 种群规模$N_{\rm P}$和各变量的上下界值, 置无效区域*IR*为空, 迭代代数$g=0$, 在各变量定义域范围内随机生成初始种群$P=\{x_1,x_2,\cdots,x_{N_{\rm P}}\}$, 根据式(6)对单位单纯形区域的各顶点作转换得到点$x^1,x^2,\cdots,x^{N+1}$, 并根据式(10)计算各点的支撑向量$l^1,l^2,\cdots,l^{N+1}$;

Step 2 找出当前种群中的最优个体$x_{\rm best}$和最差个体$x_{\rm worst}$, 如果满足终止条件(如$|f(x_{\rm best})-f(x_{\rm worst})|\leqslant\epsilon$), 则退出;

Step 3 对于每一个目标个体$x_i\in P$, 经过变异、交叉生成新个体$x_{\rm trial}$;

Step 4 通过选择策略来替换种群P中的个体, 对于每一个新个体$x_{\rm trial}$, 通过如下操作来决定它是否替换其对应的目标个体x_i;

Step 4.1 以支撑矩阵$L=\{l^1,l^2,\cdots,l^{N+1}\}$为根建立$n$叉树来保存各下界估计值;

Step 4.2 找出离$x_{\rm trial}$最近的m个个体, 根据式(5)作转换后, 再根据式(10)求支撑向量, 并根据式(12)–(13)更新树;

Step 4.3 找出包含$x_{\rm trial}$的树叶子节点Node, 如果$x_{\rm trial}$被包含在无效区域*IR*中, 则保留x_i不变, 并转到Step 4.8, 否则转到Step 4.4;

Step 4.4 根据式(8)计算出$x_{\rm trial}$的下界估计值$y_{\rm trial}$, 如果$y_{\rm trial}$大于目标个体x_i的函数值, 则保留x_i不变并转到Step 4.5, 否则转到Step 4.6;

Step 4.5 根据式(16)计算出Node节点所对应的下界估计区域的极小值$d_{\min}$, 如果$d_{\min}$大于当前种群的最优值$f(x_{\rm best})$, 则将Node区域视为无效区域, 且加入*IR*中, 并转到Step 4.8;

Step 4.6 计算$x_{\rm trial}$的目标函数值, 如果小于x_i的目标函数值, 则$x_{\rm trial}$替换x_i并转到Step 4.7, 否则转到Step 4.8;

Step 4.7 根据式(15)计算出Node节点所对应的下界估计区域的极小解$x'^*_{\min}$, 根据式(6)计算出$x^*_{\min}$; 如果$x^*_{\min}$对应的目标函数值小于$x_{\rm trial}$的目标函数值, 则$x^*_{\min}$取代$x_{\rm trial}$, 并转到Step 4.8;

Step 4.8 删除树, 并转到Step 2;

Step 5 设置$g=g+1$, 并转到Step 2.

注 1 本文在Step 4.2中只对$x_{\rm trial}$最近的两个个体构建支撑面, 并以n叉树的形式来保存各下界估计值, 事实上, 树的规模只有两个节点, 读者也可以以其他形式来实现对下界估计值的保存; Step 4.2中树的更新过程及Step 4.3中找出包含$x_{\rm trial}$的树叶子节点的方法参见文献[21]; 为了降低空间复杂度, 在更新环节结束后, 对支撑面作删除操作, 只保留了无效区域的虚拟支撑面, 用于判断新个体是否落在其中.

3.4 复杂度分析(Complexity analysis)

根据算法流程进行时间复杂度分析, Step 1, Step 4.2中构建支撑向量的时间复杂度为$\mathrm{O}(N+1)$, Step 4.2中找出离$x_{\rm trial}$最近的m个个体的时间复杂度为$\mathrm{O}(N_{\rm P})$, Step 4.3中找出包含$x_{\rm trial}$的树叶子节点的时间复杂度为$\log T$, 其中T为n叉树的节点数目, 因此, 整个算法的平均时间复杂度为$\mathrm{O}(N+1+N_{\rm P}+\log T)$. DERL算法中锦标赛机制的时间复杂度为$\mathrm{O}(t^2)$, t为锦标赛机制中个体的个数; DELB算法中引入反射和收缩算子的时间复杂度为$\mathrm{O}(N_{\rm P}+2N)$, SaDE算法中参数及策略自适应机制的时间复杂度为$\mathrm{O}(N_{\rm P})$. 因此, 相比于其他算法, 抽象凸估计选择策略的引入并没有显著增加算法的时间复杂度.

4 数值实验与分析(Numerical experiments and analysis)

4.1 测试函数与参数设置(Benchmark functions and parameter settings)

应用12个典型的标准测试函数来保证数值实验的客观性, 维数$N=2\sim30$, 表1给出了各测试函数参数, 具体表达式参见文献[25]. 12个函数中包含了4个单模态函数(f_1-f_4)和8个多模态函数(f_5-f_{12}), 多模态函数局部最优解的个数随着维数的增大而增加. 函数f_4和f_9的最优解为$x=(1,\cdots,1)$, 函数f_7的最优解为$x=(420.97,\cdots,420.97)$, 函数f_8的最优解为$x=(-1,\cdots,-1)$, 其余函数的最优解均在原点.

表 1 12个标准测试函数

Table 1 12 benchmark functions

函数名	维数	搜索范围	全局最小值
f_1: Sphere	10, 30	$(-100,100)^N$	0
f_2: Exponential	10, 30	$(-1,1)^N$	-1
f_3: Zakharov	5, 10	$(-5,10)^N$	0
f_4: Rosenbrock	2, 4	$(-2,2)^N$	0
f_5: Griewank	10, 30	$(-600,600)^N$	0
f_6: Schaffer2	10, 30	$(-100,100)^N$	0
f_7: Schwefel	10, 30	$(-500,500)^N$	$-418.983N$
f_8: LM1	10, 30	$(-10,10)^N$	0
f_9: LM2	10, 30	$(-5,5)^N$	0
f_{10}: Ackley	5, 10	$(-30,30)^N$	0
f_{11}: Rastrigin	5, 10	$(-5.12,5.12)^N$	0
f_{12}: CM	2, 4	$(-1,1)^N$	$-0.1N$

为了验证所提算法的优势, 本文选用标准DE和DERL, DELB, SaDE 3种改进算法进行比较分析; 同时, 鉴于比较的合理性和公平性, 并验证抽象凸估计选择策略的优势, 而不考虑其他因素的影响, 将DE, DERL, DELB和DEUS算法的增益常数F和交叉概率$C_{\rm R}$均设置为0.5, 函数f_1-f_3种群规模$N_{\rm P}$为20, 其余函数为30. 所有实验均独立运行30次.

4.2 计算代价与可靠性(Computational cost and reliability)

选取函数评价次数(function evaluations, FES)和成功率(success rate, SR)两个指标来验证DEUS算法在计算代价和可靠性方面的优势, 其中SR为算法的成功运行次数与总运行次数之比, 算法终止条件为$|f(x_{\text{best}})-\text{Opti}| \leqslant 10^{-5}$, “Opti”为函数的全局最小值. 结果见表2, 其中最优结果通过加粗标出.

表 2 函数评价次数和成功率

Table 2 Function evaluations and success rates

Fun	N	DE		DERL		DELB		SaDE		DEUS	
		FES	SR	FES	SR	FES	SR	FES	SR	FES	SR
f_1	30	12094	1.00	8765	0.90	13747	1.00	20448	1.00	**8542**	1.00
	10	4020	1.00	3089	1.00	3257	1.00	3716	1.00	**2147**	1.00
f_2	30	6733	1.00	4948	1.00	6560	0.83	8516	1.00	**4763**	1.00
	10	2139	1.00	1672	1.00	1716	1.00	2637	1.00	**1544**	1.00
f_3	10	8663	1.00	5824	0.97	6828	1.00	12739	1.00	**3926**	1.00
	5	2212	1.00	1598	1.00	1889	1.00	1941	1.00	**1248**	1.00
f_4	4	8339	0.60	7475	0.47	**7083**	1.00	7941	0.83	7288	0.63
	2	2449	0.93	1964	0.87	2231	0.97	2079	0.90	**1913**	0.93
f_5	30	22346	0.97	15765	0.90	13459	0.70	16824	0.57	**12156**	1.00
	10	21551	0.97	15369	0.90	18550	0.97	16703	0.97	**13103**	0.97
f_6	30	81375	1.00	57841	0.93	**49560**	0.40	50927	0.23	51215	1.00
	10	22567	1.00	**17396**	1.00	18003	1.00	19431	0.97	18513	1.00
f_7	30	65366	0.87	48683	0.73	63341	0.90	**44349**	0.83	50106	0.97
	10	9210	0.97	7310	0.83	8268	0.93	8097	0.97	**5206**	1.00
f_8	30	13294	1.00	9510	0.97	8787	1.00	9305	1.00	**6019**	1.00
	10	4248	1.00	3182	1.00	3373	1.00	3543	1.00	**2664**	1.00
f_9	30	13396	1.00	9530	1.00	8053	0.83	10125	0.73	**7225**	1.00
	10	4121	1.00	3093	1.00	3191	1.00	3494	1.00	**1736**	1.00
f_{10}	10	9113	1.00	6866	1.00	6939	0.97	7192	1.00	**6356**	1.00
	5	4828	1.00	3740	1.00	4179	1.00	4408	1.00	**3316**	1.00
f_{11}	10	23038	1.00	17306	0.93	22269	0.97	11514	1.00	**5618**	1.00
	5	6062	1.00	4681	1.00	5754	1.00	5730	1.00	**3002**	1.00
f_{12}	4	1731	1.00	1374	1.00	1557	1.00	1737	1.00	**406**	1.00
	2	804	1.00	643	1.00	767	1.00	836	1.00	**443**	1.00
AVE		14571	0.971	10734	0.933	11640	0.936	11426	0.917	**9102**	**0.979**

对比表2中的数据可以看出, DEUS算法在计算代价和可靠性方面, 除了函数f_4, f_6和f_7–30维稍逊于其他算法外, 其余函数均优于其他4种算法. 对于函数f_6和f_7–30维, DEUS算法虽然函数评价次数稍高, 但是成功率均最高. 此外, 从12个测试函数的平均结果来看, DEUS算法计算代价最小, 平均函数评价次数为9102, DE, DERL, DELB和SaDE算法分别为14571, 10734, 11640和11426, 也就是说, DEUS算法相对于DE, DERL, DELB和SaDE算法分别节省了37.5%, 15.2%, 21.8%和20.3%; 而且DEUS算法的可靠性最高, 平均成功率为0.979, DE, DERL, DELB和SaDE算法分别为0.971, 0.933, 0.936和0.917. DE算法虽然可靠性仅次于DEUS算法, 但计算代价较大, 其他3种改进算法虽然计算代价得到了改善, 但是可靠性较低, 而采用抽象凸估计选择策略的DEUS算法不仅计算代价得到了改善, 而且能够保证算法的可靠性.

为了进一步说明DEUS算法在计算代价和可靠性方面的整体性能, 继续绘制12个测试函数的成功表现(success performance, SP=平均函数评价次数/成功率)归一化经验分布图[11]来进行比较. 首先, 根据表2计算出5种算法对于每个测试函数的SP值,

然后通过用各测试函数所有的SP值除以最好算法的SP值对SP进行归一化处理, 图中SP值最小而经验分布值最大为最好, 即第1个到达图形顶部的算法为最好算法. 如图3所示, DEUS算法在计算代价和可靠性方面的整体性能明显优于其他4种算法, DERL算法次之. 此外, 需要说明的是, 为了图形的清晰直观, 图3中只截取了经验分布图的一部分.

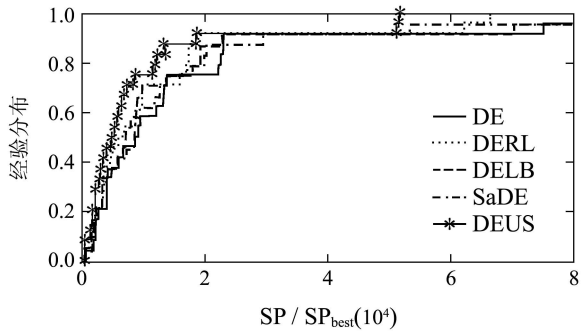

图 3 成功表现归一化经验分布图

Fig. 3 The empirical distribution of normalized success performance

4.3 优化结果分析(Optimization results analysis)

以函数评价次数为终止条件, 从优化结果的平均值(mean)和标准偏差(standard deviation, Std)两方面对5种算法30次独立运行所得的结果进行比较分析; 同时, 为了验证所提算法相对于其他4种算法是否有显著性优势, 选用Wilcoxon Signed Rank Test[26]对优化结果进行非参数假设检验, 显著性水平为0.05, 结果见表3−4. 表3和表4分别给出了单模态函数和多模态函数优化结果的两项性能数据, 其中: 最优结果通过加粗标出, “+”表示所提算法显著优于所比算法, “≈”表示所提算法与所比算法没有显著性差异, “−”表示所提算法显著差于所比算法.

从表3可以看出, DEUS算法在对4个单模态函数(f_1-f_4)的8个问题进行优化求解时, 在给定的目标函数评价次数内, 30次独立运行所求得的最优结果的平均值及标准偏差, 除了函数f_4稍逊于DELB算法外, 其余函数优化结果的两项性能均显著优于其他4种算法, 且DEUS算法的平均值优于其他4种算法中的最优算法好几个数量级, 如f_1–30维问题优于DE算法5个数量级, f_3–10维问题优于DELB算法4个数量级, 而且只有DEUS算法对于函数f_2的平均值等于全局最优值; 另外, 函数f_1-f_3的标准偏差最小, 说明对于绝大多数单模态函数, DEUS算法比较稳定, 解的质量较高. 此外, 从表3的最后一行可以看出, 在显著性方面, DEUS算法分别显著优于DE, DERL, DELB和SaDE算法7个, 4个, 5个和6个问题, 对于其余问题, 除了函数f_4外, 虽然与其他4种算法相比没有显著性差异, 但结果均优于其他4种算法.

表 3 单模态函数的优化结果性能表

Table 3 The performance of optimization results on unimodal functions

Fun	N	FES	指标	DE		DERL		DELB		SaDE		DEUS
f_1	30	15000	Mean	1.80E–08	+	6.27E–03	+	2.70E–05	+	7.14E–03	+	**1.55E–13**
			Std	1.65E–08		2.99E–02		1.21E–04		1.90E–02		**1.35E–13**
	10	5000	Mean	1.14E–07	+	3.58E–11	+	1.69E–10	+	1.00E–06	+	**1.27E–12**
			Std	1.14E−07		4.83E−11		3.66E−10		3.38E−06		**1.31E−12**
f_2	30	7000	Mean	–0.999993	+	–0.999977	≈	–0.999977	≈	–0.999809	+	**–1**
			Std	2.75E–06		1.24E–04		1.16E–05		4.59E–04		**0.00E–00**
	10	2000	Mean	–0.999985	+	–0.999999	+	–0.999999	+	–0.999991	+	**–1**
			Std	1.08E–05		8.50E–07		1.79E–06		9.02E–06		**0.00E–00**
f_3	10	10000	Mean	7.69E–07	+	1.42E–05	+	2.91E–08	+	1.10E–01	+	**2.40E–12**
			Std	9.30E–07		7.63E–05		1.10E–07		5.90E–01		**4.30E–12**
	5	3000	Mean	5.64E–08	+	6.16E–11	≈	2.44E–09	+	7.00E–08	+	**1.79E–11**
			Std	7.26E–08		1.71E–10		4.22E–09		2.10E–07		**3.78E–11**
f_4	4	10000	Mean	3.00E–01	≈	3.52E–01	≈	**7.26E–07**	–	8.86E–02	≈	8.55E–02
			Std	7.26E–01		9.49E–01		**3.21E–06**		6.47E–01		1.78E–01
	2	3000	Mean	6.72E–05	+	1.27E–02	≈	**1.28E–05**	–	2.98E–04	≈	4.81E–05
			Std	3.30E–04		5.84E–02		**6.72E–05**		1.60E–03		1.85E–04
+/≈/−				7/1/0		4/4/0		5/1/2		6/2/0		

表 4 多模态函数的优化结果性能表

Table 4 The performance of optimization results on multimodal functions

Fun	N	FES	指标	DE		DERL		DELB		SaDE		DEUS
f_5	30	25000	Mean	4.31E−06	+	9.04E−04	+	3.93E−03	+	1.50E−02	+	**1.49E−14**
			Std	1.04E−05		2.86E−03		9.35E−03		2.06E−02		**7.66E−14**
	10	25000	Mean	2.98E−03	+	5.69E−03	+	2.90E−03	≈	1.72E−03	≈	**8.08E−05**
			Std	1.22E−02		7.34E−03		1.41E−02		5.74E−03		**4.42E−04**
f_6	30	9000	Mean	8.57E−07	+	3.10E−03	+	1.19E−01	+	1.15E+00	+	**1.10E−15**
			Std	5.50E−07		1.25E−02		3.54E−01		2.26E+00		**7.53E−16**
	10	25000	Mean	1.37E−06	+	**1.57E−09**	−	2.90E−09	≈	4.64E−08	+	3.57E−09
			Std	7.50E−07		**8.24E−10**		2.23E−09		3.30E−08		1.93E−09
f_7	30	70000	Mean	−12412.8	+	−12502.4	+	−12466.3	≈	−12545.8	+	**−12565.6**
			Std	4.86E+02		8.62E+01		3.39E+02		4.82E+01		**2.16E+01**
	10	10000	Mean	−4185.88	≈	−4174.04	≈	−4181.93	≈	−4189.83	≈	**−4189.83**
			Std	2.16E+01		4.10E+01		3.00E+01		1.01E−05		**0.00E−00**
f_8	30	15000	Mean	2.38E−06	+	2.30E−05	+	1.76E−07	+	3.80E−08	+	**2.23E−11**
			Std	1.73E−06		1.26E−04		9.60E−07		6.51E−08		**1.72E−11**
	10	5000	Mean	2.99E−07	+	1.53E−09	+	4.05E−09	+	3.27E−08	+	**4.63E−10**
			Std	1.70E−07		1.45E−09		9.55E−09		3.96E−08		**4.56E−10**
f_9	30	15000	Mean	1.08E−06	+	1.46E−09	+	3.23E−10	+	7.01E−04	+	**1.73E−10**
			Std	6.75E−07		1.43E−09		8.93E−10		3.84E−03		**1.02E−10**
	10	5000	Mean	2.65E−07	+	7.49E−10	+	1.29E−09	+	2.45E−08	+	**2.46E−11**
			Std	2.66E−07		4.86E−10		9.79E−10		2.64E−08		**2.07E−11**
f_{10}	10	10000	Mean	1.10E−06	+	3.14E−09	+	3.40E−09	+	2.99E−08	+	**1.02E−09**
			Std	6.35E−07		1.79E−09		2.12E−09		2.23E−08		**4.46E−10**
	5	5000	Mean	2.99E−06	+	2.29E−08	+	2.13E−07	+	1.35E−06	+	**2.87E−09**
			Std	1.44E−06		1.66E−08		1.55E−07		8.20E−07		**2.47E−09**
f_{11}	10	25000	Mean	2.63E−05	+	3.32E−02	≈	1.31E−05	+	3.32E−02	+	**0.00E−00**
			Std	1.20E−04		1.82E−01		7.18E−05		1.82E−01		**0.00E−00**
	5	8000	Mean	1.53E−10	+	3.32E−02	≈	1.31E−12	+	6.70E−11	+	**0.00E−00**
			Std	5.59E−10		1.82E−01		4.29E−12		1.42E−10		**0.00E−00**
f_{12}	4	1500	Mean	−0.399943	+	−0.399999	+	−0.399987	+	−0.399819	+	**−0.4**
			Std	4.87E−05		1.87E−06		1.25E−05		2.17E−04		**0.00E−00**
	2	800	Mean	−0.199989	+	−0.199999	+	−0.199992	+	−0.199987	+	**−0.2**
			Std	1.25E−05		5.83E−07		1.14E−05		1.64E−05		**0.00E−00**
		+/≈/−		15/1/0		12/3/1		12/4/0		14/2/0		

表4的对比结果表明, 在给定的目标函数评价次数内, DEUS算法在对8个多模态函数(f_5-f_{12})的16个问题进行优化求解时, 在优化结果的平均值和标准偏差方面, 除了f_6–10维问题稍逊于DERL算法外, 其余问题均优于其他4种算法, 而且与单模态函数一样, DEUS算法的平均值也优于其他4种算法中的最优算法好几个数量级, 如f_5–30维问题优于DE算法8个数量级, f_6–30维问题优于DE算法8个数量级, f_8–30维问题优于SaDE算法3个数量级等, 尤其对于函数f_{11}, 只有DEUS算法的平均值等于函数的全局最优值, 而其他算法与全局最优值还相差甚远; 其次, DEUS算法的标准偏差也比其他算法小很多, 由此说明, 对于绝大多数多模态函数, DEUS算法也比较稳定, 解的质量也较高. 此外, 从表4最后一行可以看出, 在显著性方面, DEUS算法的优化结果分别显著优于DE, DERL, DELB和SaDE算法15个、12个、12个和14个问题, 对于其余问题, 除了f_6–10维问题外, 虽然与其他4种算法相比结果没有显著性差异, 但是优化结果均优于其他4种算法.

4.4 平均收敛特性(Mean convergence characteristics)

为了验证DEUS算法在收敛速度方面的优势, 在给定的迭代代数内, 基于各算法30次独立运行的平均结果, 对每个测试函数选取一个维数, 以算法迭代代数为横坐标, 平均函数值为纵坐标, 绘制5种算法的平均收敛特性图来进行比较分析, 结果如图4所示. 需要说明的是, 为了图形的清晰直观, 对于最优值为0的函数, 纵坐标均取函数值的对数.

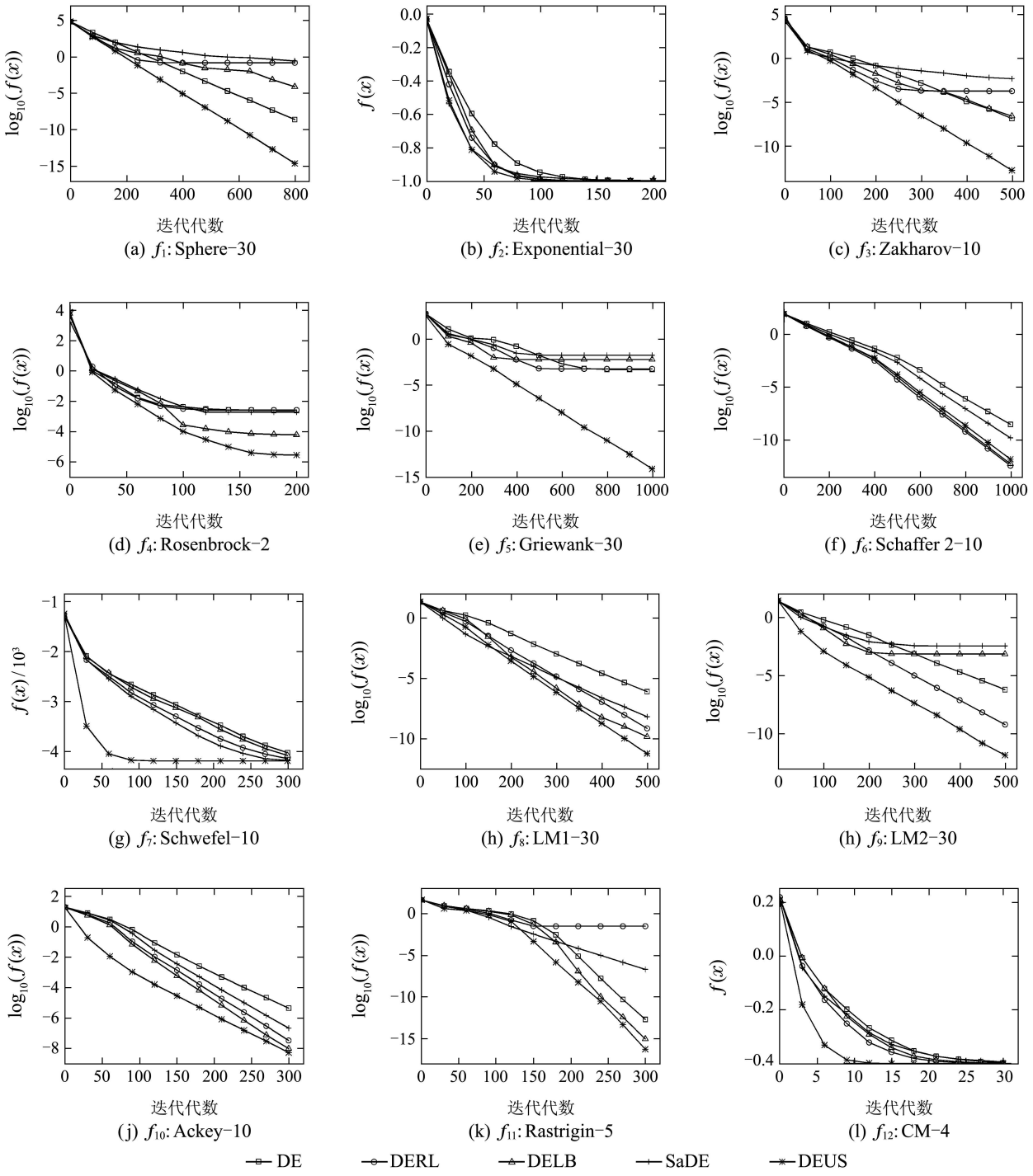

图 4 平均收敛特性曲线图

Fig. 4 The curve graph of mean convergence characteristics

通过图4可以看出, 由于DEUS算法根据下界估计区域的极值信息系统排除部分无效区域, 并借助线性拟凸包络的广义下降方向有效地实现局部增强, 有效加快了收敛速度, 所以在对12个测试问题进行求解时, 除了函数f_6的收敛速度稍慢于DERL和DELB算法外, 其余11个测试问题的收敛速度均优于DE, DERL, DELB及SaDE算法. 从图4(b), 4(c), 4(h)和4(k)可以看出, 在对函数f_2, f_3, f_8和f_{11}进行优化求解时, 虽然开始时DEUS算法的收敛速度稍慢于SaDE算法, 但是很快就超过了其他4种算法, 并且非常稳定地向着全局最优方向收敛; 其次, 如图4(d)所示, 对于函数f_4, 虽然5种算法都陷入了局部最优解, 但是DEUS算法的收敛速度最快; 而且, 从图4(e)和4(g)可以看出, 对于函数f_5和f_7, 其他4

种算法均陷入了局部最优解, 只有DEUS算法能够成功求解. 总体来说, DEUS算法不仅收敛速度快, 而且不易陷入局部最优.

5 结论(Conclusions)

本文提出了一种基于抽象凸估计选择策略的差分进化算法, 在DE算法中, 通过引入抽象凸估计选择策略, 来减小算法的计算代价, 加快算法的收敛速度, 同时提高算法的可靠性. 事实上, 在选择过程中, 抽象凸下界支撑面的建立是为了获取新个体的下界信息, 从而利用下界信息来指导更新, 并系统排除部分无效区域, 同时实现局部增强, 因此, 只需对新个体的邻近个体建立抽象凸下界支撑面, 而且一旦选择操作完成后立即删除, 所以, 抽象凸估计选择策略的引入并没有带来较大的复杂度. 12个典型测试函数的实验结果表明, DEUS算法不仅可以以廉价的计算代价求得质量较高的解, 而且可靠性较高, 收敛速度较快, 是一种有效地全局优化算法, 且提出的抽象凸估计选择策略可以广泛应用到其他群体算法中.

参考文献(References):

[1] FLOUDAS C A, GOUNARIS C E. A review of recent advances in global optimization [J]. *Journal of Global Optimization*, 2009, 45(1): 3 – 38.

[2] STORN R, PRICE K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces [J]. *Journal of Global Optimization*, 1997, 11(4): 341 – 359.

[3] GOLDBERG D E, HOLLAND J H. Genetic algorithms and machine learning [J]. *Machine Learning*, 1988, 3(2): 95 – 99.

[4] RECHENBERG I. *Evolutions Strategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution* [M]. Stuttgart: Fromman-Holzboog, 1973.

[5] FOGL L, OWNES J, WALSH M. *Artificial Intelligence Through Simulated Evolution* [M]. New York: Wiley, 1966.

[6] KENNEDY J. *Particle Swarm Optimization* [M]. New York: Springer, 2010.

[7] DAS S, SUGANTHAN P N. Differential evolution: a survey of the state-of-the-art [J]. *IEEE Transactions on Evolutionary Computation*, 2011, 15(1): 4 – 31.

[8] 叶洪涛, 罗飞, 许玉格. 解决多目标优化问题的差分进化算法研究进展 [J]. 控制理论与应用, 2013, 30(7): 922 – 928.
(YE Hongtao, LUO Fei, XU Yuge. Differential evolution for solving multi-objective optimization problems: a survey of the state-of-the-art [J]. *Control Theory & Applications*, 2013, 30(7): 922 – 928.)

[9] STOEAN C, PREUSS M, STOEAN R, et al. Multimodal optimization by means of a topological species conservation algorithm [J]. *IEEE Transactions on Evolutionary Computation*, 2010, 14(6): 842 – 864.

[10] KAELO P, ALI M M. A numerical study of some modified differential evolution algorithms [J]. *European Journal of Operational Research*, 2006, 169(3): 1176 – 1184.

[11] QIN A K, HUANG V L, SUGANTHAN P N. Differential evolution algorithm with strategy adaptation for global numerical optimization [J]. *IEEE Transactions on Evolutionary Computation*, 2009, 13(2): 398 – 417.

[12] GAO Y, WANG Y J. A memetic differential evolutionary algorithm for high dimensional functions' optimization [C] *//The 3rd International Conference on Natural Computation (ICNC)*. Haikou: IEEE, 2007, 4: 188 – 192.

[13] CHIOU J P, WANG F S. Hybrid method of evolutionary algorithms for static and dynamic optimization problems with application to a fed-batch fermentation process [J]. *Computers & Chemical Engineering*, 1999, 23(9): 1277 – 1291.

[14] BHATTACHARYA A, CHATTOPADHYAY P K. Hybrid differential evolution with biogeography-based optimization for solution of economic load dispatch [J]. *IEEE Transactions on Power Systems*, 2010, 25(4): 1955 – 1964.

[15] ELSAYED S M, SARKER R A, ESSAM D L. An improved self-adaptive differential evolution algorithm for optimization problems [J]. *IEEE Transactions on Industrial Informatics*, 2013, 9(1): 89 – 99.

[16] WANG F S, JING C H, TSAO G T. Fuzzy-decision-making problems of fuel ethanol production using a genetically engineered yeast [J]. *Industrial & Engineering Chemistry Research*, 1998, 37(8): 3434 – 3443.

[17] SONG X M, TANG L X. A novel hybrid differential evolution-estimation of distribution algorithm for dynamic optimization problem [C] *//IEEE Congress on Evolutionary Computation (CEC)*. Cancun: IEEE, 2013: 1710 – 1717.

[18] RAHNAMAYAN S, TIZHOOSH H R, SALAMA M M A. Opposition-based differential evolution [J]. *IEEE Transactions on Evolutionary Computation*, 2008, 12(1): 64 – 79.

[19] ROGERS A, PRUGEL-BENNETT A. Genetic drift in genetic algorithm selection schemes [J]. *IEEE Transactions on Evolutionary Computation*, 1999, 3(4): 298 – 303.

[20] RUBINOV A M. *Abstract Convexity and Global Optimization, of Nonconvex Optimization and Its Applications* [M]. New York: Springer, 2000.

[21] BELIAKOV G. Geometry and combinatorics of the cutting angle method [J]. *Optimization*, 2003, 52(4/5): 379 – 394.

[22] BAGIROV A M, RUBINOV A M. Global minimization of increasing positively homogeneous functions over the unit simplex [J]. *Annals of Operations Research*, 2000, 98(1/4): 171 – 187.

[23] 邓勇跃, 张贵军. 基于局部抽象凸支撑面的多模态优化算法 [J]. 控制理论与应用, 2014, 31(4): 458 – 466.
(DENG Yongyue, ZHANG Guijun. Multimodal optimization based on local abstract convexity support hyperplanes [J]. *Control Theory & Applications*, 2014, 31(4): 458 – 466.)

[24] 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 Guijun, HE Yangjun, GUO Haifeng, et al. A differential evolution algorithm for multimodal optimization based on abstract convex underestimation [J]. *Journal of Software*, 2013, 24(6): 1177 – 1195).

[25] ALI M M, KHOMPATRAPORN C, ZABINSKY Z B. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems [J]. *Journal of Global Optimization*, 2005, 31(4): 635 – 672.

[26] CORDER G W, FOREMAN D I. *Nonparametric Statistics for Non-Statisticians: A Step-By-Step Approach* [M]. Hoboken: John Wiley & Sons, 2009.

作者简介:

周晓根　(1987–), 男, 博士研究生, 主要研究领域为智能优化, E-mail: zhouxiaogen53@126.com;

张贵军　(1974–), 男, 博士, 教授, 主要研究领域为智能信息处理、全局优化理论及算法设计、生物信息学, E-mail: zgj@zjut.edu.cn;

梅　珊　(1990–), 女, 硕士研究生, 主要研究领域为智能优化, E-mail: mayshan1990@hotmail.com;

明　洁　(1990–), 女, 硕士研究生, 主要研究领域为智能交通系统和优化调度, E-mail: zzzzmj@126.com.