文章编号: 1001-0920 (2015) 06-1116-05 DOI: 10.13195/j.kzyjc.2014.0470

基于抽象凸下界估计的群体全局优化算法

张贵军, 周晓根

(浙江工业大学 信息工程学院，杭州 310023)

摘 要: 针对确定性全局优化算法极高的计算复杂度以及随机性全局优化算法可靠性较低的问题, 在群体进化算法框架下, 结合抽象凸理论, 提出一种基于抽象凸下界估计的群体全局优化算法. 首先, 对整个初始群体构建抽象凸下界估计松弛模型; 然后, 利用不断收紧的下界估计信息安全排除部分无效区域, 并指导种群更新, 同时借助支撑面的下降方向作局部增强; 最后, 根据进化信息更新支撑面. 数值实验结果表明了所提出算法的有效性.

关键词: 进化算法； 下界估计； 全局优化； 支撑向量； 抽象凸

中图分类号: TP391 **文献标志码:** A

Population-based global optimization algorithm using abstract convex underestimate

ZHANG Gui-jun, *ZHOU Xiao-gen*

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

Abstract: To solve the problem of high computation complexity in deterministic global optimization algorithms and low success ratio in stochastic global optimization algorithms, a population-based global optimization algorithm using abstract convex underestimate is proposed. The proposed algorithm combines the abstract convexity theory within the framework of population evolutionary algorithms. The first step of the algorithm is to construct the abstract convex underestimate relaxed model for the whole initial population. Then, relevant tightening underestimate information is used to safely eliminate invalid regions and to guide the population updating. Additionally, descent directions of supporting hyperplanes are employed for local enhancement. Finally, evolutionary information helps to update supporting hyperplanes. Numerical experiment results show the effectiveness of the proposed algorithm.

Keywords: evolutionary algorithms； underestimate； global optimization； support vector； abstract convex

0 引 言

在实际工程应用中, 许多优化问题往往需要快速得到某一个全局最优解. 基于梯度的拟牛顿法、共轭梯度法等传统算法[1], 以及 Nelder-Mead[2]、Hooke-Jeeves[3] 等直接搜索算法本质上都属于一类局部搜索算法, 解的质量直接取决于起始点的选择, 对于一些复杂的优化问题, 这些算法基本上不可能得到问题的全局最优解.

总体上, 全局优化算法可分为确定性算法和随机性算法. 确定性算法利用问题的解析性质产生确定性的有限或无限点序列使其收敛于全局最优解, 同时获取相应的上下界信息. 典型的算法有凸分析[4]、分支定界法[5]以及抽象凸分析[6-7]等. 这些算法通常依赖于待解问题的先验知识 (如 Lipshitz 常数), 可靠性较高, 但是极高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[8]. 在实际工程中广泛应用的随机性算法利用概率机制而非确定的点列来描述迭代过程. 常见的算法有遗传算法 (GA)[9]、差分进化算法 (DE)[10-11]以及粒子群算法 (PSO)[12]等, 这些算法属于元启发式算法, 虽然在某些情况下能得到问题的全局最优解, 但是极容易陷入局部最优解, 可靠性较低[13].

针对上述问题, 权衡确定性算法与随机性算法的

收稿日期: 2014-04-02；**修回日期:** 2014-08-17.

基金项目: 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项目(20120811)；杭州市产学研合作基金项目(20131631E31).

作者简介: 张贵军(1974−), 男, 教授, 博士生导师, 从事智能信息处理、优化理论及算法设计、生物信息学等研究；周晓根(1987−), 男, 博士生, 从事智能优化的研究.

缺陷, 并综合利用两者优势, 本文提出一种基于抽象凸下界估计的群体全局优化算法 (ACUP). 在群体进化算法框架下, 首先对初始群体建立抽象凸下界支撑面, 进而利用不断收紧的下界信息指导种群更新, 即利用下界估计值与目标个体的函数值比较决定是否需要对新个体作函数评价, 有效减少了函数评价次数; 然后, 根据下界估计区域的极值信息安全排除部分无效区域 (即在该区域不包含全局最优解), 不仅提高了算法的可靠性, 而且加快了收敛速度; 同时借助支撑面的下降方向作局部增强, 进一步加快算法的收敛速度; 最后, 根据进化信息更新支撑面以降低算法的计算复杂度.

1　理论基础

1.1　模型转换

考虑如下全局优化问题:

$$\min\ f(\boldsymbol{x}),\ \boldsymbol{x}\in[\boldsymbol{a},\boldsymbol{b}]\subset R^N. \tag{1}$$

其中: $\boldsymbol{x}=[x_1,x_2,\cdots,x_N]^{\mathrm{T}}$ 为边界约束可行域 $[\boldsymbol{a},\boldsymbol{b}]$ 上的 N 维连续优化变量, $\boldsymbol{a}=[a_1,a_2,\cdots,a_N]^{\mathrm{T}}$ 和 $\boldsymbol{b}=[b_1,b_2,\cdots,b_N]^{\mathrm{T}}$ 分别为可行域的下界和上界向量, $f(\cdot)$ 为定义在可行域 $[\boldsymbol{a},\boldsymbol{b}]$ 上的目标函数.

对原优化变量 $\boldsymbol{x}$ 作如下线性变换:

$$\begin{cases} x'_i\equiv(x_i-a_i)\Big/\sum\limits_{i=1}^{N}(b_i-a_i), \\ x'_{N+1}\equiv 1-\sum\limits_{i=1}^{N}x'_i, \end{cases} \quad i=1,2,\cdots,N. \tag{2}$$

其中 $\boldsymbol{x}'=[x'_1,x'_2,\cdots,x'_{N+1}]^{\mathrm{T}}$ 为定义在单位单纯形区域 $S\equiv\left\{\boldsymbol{x}'\in R^{N+1},x'_i\geqslant 0,\sum\limits_{i=1}^{N+1}x'_i=1\right\}$ 中的线性转换变量.

对式 (2) 进行反变换, 有

$$x_i=x'_i\sum_{i=1}^{N}(b_i-a_i)+a_i,\ i=1,2,\cdots,N. \tag{3}$$

将式 (3) 代入 (1), 可得

$$\min\ \bar{f}(\boldsymbol{x}'),\ \boldsymbol{x}'\in S\subset R_+^{N+1}, \tag{4}$$

其中 $\bar{f}(\cdot)$ 为定义在单位单纯形区域 S 上的 $N+1$ 维目标函数.

1.2　松弛模型

假定有 K 个支撑函数, 基于式 (4) 目标函数的给定点 $(\boldsymbol{x}'^k,\bar{f}(\boldsymbol{x}'^k)),k=1,2,\cdots,K$, 函数 $\bar{f}(\cdot)$ 的松弛模型为

$$H^K(\boldsymbol{x}')=\max_{k\leqslant K}\min_{i=1,\cdots,N+1}l_i^k x'_i, \tag{5}$$

其中

$$l^k=\Big(\frac{\bar{f}(\boldsymbol{x}'^k)}{x'^k_1},\frac{\bar{f}(\boldsymbol{x}'^k)}{x'^k_2},\cdots,\frac{\bar{f}(\boldsymbol{x}'^k)}{x'^k_{N+1}}\Big), \tag{6}$$

称为支撑向量.

引理 1　设 $\exists\bar{L}>0$, 使得式 (4) 的目标函数 $\bar{f}:S\to R$ 满足

$$\bar{L}=\inf_{\boldsymbol{x}'^1\neq\boldsymbol{x}'^2}\frac{|\bar{f}(\boldsymbol{x}'^1)-\bar{f}(\boldsymbol{x}'^2)|}{\|\boldsymbol{x}'^1-\boldsymbol{x}'^2\|_1},\ \forall\boldsymbol{x}'^1,\boldsymbol{x}'^2\in S. \tag{7}$$

取 $M>2\bar{L}-\min\limits_{\boldsymbol{x}'\in S}\bar{f}(\boldsymbol{x}')$, 则 $\forall\boldsymbol{y}\in S$ 的支撑函数

$$h^y(\boldsymbol{x}')=\min_{i=1,\cdots,N+1}l_i x'_i,\ \forall\boldsymbol{x}'\in S, \tag{8}$$

满足 $h^y(\boldsymbol{x}')\leqslant\bar{f}(\boldsymbol{x}')+M,\forall\boldsymbol{x}'\in S$. 其中

$$\|\boldsymbol{x}'^1-\boldsymbol{x}'^2\|_1\equiv\max_{i=1,\cdots,N+1}|x'^1_i-x'^2_i|,$$

$$l_i=(\bar{f}(\boldsymbol{y})+M)/y_i.$$

推论 1　设 $\boldsymbol{y}^1,\boldsymbol{y}^2,\cdots,\boldsymbol{y}^K\in S$ 为式 (4) 目标函数的已知采样点, 则式 (5) 的松弛模型满足

$$\bar{f}(\boldsymbol{x}')+M\geqslant H^K(\boldsymbol{x}'),\ \forall\boldsymbol{x}'\in S, \tag{9}$$

$$\bar{f}(\boldsymbol{x}')+M=H^K(\boldsymbol{x}'),\ \forall\boldsymbol{x}'\in\{\boldsymbol{y}^1,\boldsymbol{y}^2,\cdots,\boldsymbol{y}^K\}. \tag{10}$$

注 1　引理 1 和推论 1 的证明详见文献 [14].

1.3　模型求解

根据模型转换公式 (2) 和引理 1, 将原目标函数转化为单位单纯形约束下的目标函数后, 可以利用高效模型求解算法对模型进行求解. 考虑一个含有 K 个支撑向量的集合 $\Lambda^k=\{\boldsymbol{l}^k\}_{k=1}^K$, 令 $I=\{1,2,\cdots,N+1\}$. 假设一个 $N+1$ 维支撑向量矩阵

$$L=\begin{bmatrix} l_1^{k_1} & l_2^{k_1} & \cdots & l_{N+1}^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \cdots & l_{N+1}^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_{N+1}} & l_2^{k_{N+1}} & \cdots & l_{N+1}^{k_{N+1}} \end{bmatrix}, \tag{11}$$

$H^k(\boldsymbol{x}')$ 所有的局部最优解对应于满足下列两个条件的矩阵集合:

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

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

对于满足式 (12) 和 (13) 的支撑向量矩阵 L^u, 其局优解 $\boldsymbol{x}'_{\min}(L^u)$ 和局优值 $d(L^u)$ 分别为[7]

$$\boldsymbol{x}'_{\min}(L^u)=\mathrm{diag}(L^u)/\mathrm{Trace}(L^u), \tag{14}$$

$$d(L^u)=H^K(\boldsymbol{x}'_{\min})=1/\mathrm{Trace}(L^u). \tag{15}$$

2　抽象凸下界估计群体全局优化算法

2.1　建立支撑矩阵 n 叉树

假设原目标问题的维数为 N, 首先, 根据式 (6) 计算单位单纯形区域 S 的 $N+1$ 个顶点的下界支撑向量, 并以这些支撑向量组成的支撑矩阵为根设计 n 叉树来保存下界估计值; 然后, 对初始群体的每个个体及合法的新生成个体构建下界支撑面, 并根据条件关系 (12) 和 (13) 更新树.

2.2　种群更新

首先通过对整个群体建立抽象凸下界支撑面, 从

而利用下界支撑面估计目标函数值来指导种群更新; 然后利用高效的下界极值点枚举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部分无效区域, 并借助下界支撑面的下降方向作局部增强; 最后根据更新结果更新支撑面, 使得下界估计信息向目标函数不断收紧.

以图 1 所示的一维问题为例. 假设 B 为新个体, A 为目标个体, y_u^{B} 为 B 的下界估计值, 因 y_u^{B} 大于 A 的目标值, 故无需对 B 作目标函数评价, 且保留 A 不变; 继续计算出 B 所在下界估计区域的极值 d_u, 如果 d_u 大于当前种群的最小值, 则将此区域视为无效区域. 再假设 D 为新个体且不在无效区域中, C 为目标个体, y_u^{D} 为 D 的下界估计值, 因为 y_u^{D} 小于 C 的目标值, 且 D 的函数值小于 C 的目标值, 所以 D 个体取代 C 个体. 为了加快算法的收敛速度, 继续计算出 D 个体所在下界估计区域的极值点 $E(\boldsymbol{x}_u, d(\boldsymbol{x}_u))$, 及其在目标函数曲面上对应的点 $E'(\boldsymbol{x}_u, f(\boldsymbol{x}_u))$, 因为 E' 的适应度优于 D, 所以 E' 个体取代 D 个体; 然后对 E' 个体构建下界支撑面, 若 D 的适应度优于 E', 则对 D 个体构建下界支撑面.

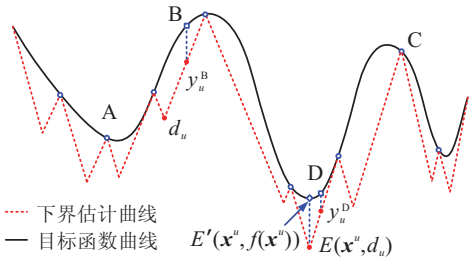

图 1 ACUP 算法中新个体更新过程

2.3 算法描述

整体算法流程如下 (极小值优化).

Step 1: 初始化. 设置常数 M, 增益常数 F, 交叉概率 C_R 和种群规模 N_P, 置无效区域 IR 为空, 代数 $g = 0$, 在各变量定义域范围内随机生成初始种群 P, 根据式 (6) 计算单位单纯形区域各顶点的支撑向量 $\boldsymbol{l}^1, \boldsymbol{l}^2, \cdots, \boldsymbol{l}^{N+1}$, 以支撑矩阵 $L = \{\boldsymbol{l}^1, \boldsymbol{l}^2, \cdots, \boldsymbol{l}^{N+1}\}^{\mathrm{T}}$ 为根建立 n 叉树.

Step 2: 根据式 (2) 对初始种群中的每个个体 x_i 进行转换, 由式 (6) 计算支撑向量, 并根据条件关系 (12) 和 (13) 更新树.

Step 3: 找出当前种群中的最优个体 $\boldsymbol{x}_{\mathrm{best}}$ 和最差个体 $\boldsymbol{x}_{\mathrm{worst}}$, 如果满足终止条件 (如 $|f(\boldsymbol{x}_{\mathrm{best}}) - f(\boldsymbol{x}_{\mathrm{worst}})| \leqslant \epsilon$), 则退出.

Step 4: 对于每个目标个体 $\boldsymbol{x}_i \in P$, 经过变异、交叉生成新个体 $\boldsymbol{x}_{\mathrm{trial}}$.

Step 5: 种群更新. 对于每个新个体 $\boldsymbol{x}_{\mathrm{trial}}$, 通过如下操作来决定它是否替换其对应的目标个体 $\boldsymbol{x}_i$:

Step 5.1: 根据式 (2) 对 $\boldsymbol{x}_{\mathrm{trial}}$ 进行转换, 判断其是否在无效区域 IR 中, 如果 $\boldsymbol{x}_{\mathrm{trial}}$ 在无效区域中, 则保留 $\boldsymbol{x}_i$ 并转 Step 3, 否则转 Step 5.2;

Step 5.2: 从 n 叉树中找出包含 $\boldsymbol{x}_{\mathrm{trial}}$ 的树叶子节点, 由式 (5) 计算出 $\boldsymbol{x}_{\mathrm{trial}}$ 的下界估计值 y_u^{trial}, 如果 y_u^{trial} 大于目标个体 $\boldsymbol{x}_i$ 的目标函数值, 则保留 $\boldsymbol{x}_i$ 不变并转 Step 5.3, 否则转 Step 5.4;

Step 5.3: 由式 (15) 计算出 $\boldsymbol{x}_{\mathrm{trial}}$ 所在的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 $\boldsymbol{x}_{\mathrm{trial}}$ 所在的区域视为无效区域, 且加入 IR 中, 并转 Step 3;

Step 5.4: 计算 $\boldsymbol{x}_{\mathrm{trial}}$ 的目标函数值, 如果小于 $\boldsymbol{x}_i$ 的目标函数值, 则 $\boldsymbol{x}_{\mathrm{trial}}$ 替换 $\boldsymbol{x}_i$ 并转 Step 5.5, 否则转 Step 3;

Step 5.5: 由式 (14) 计算出 $\boldsymbol{x}_{\mathrm{trial}}$ 所在的下界估计区域的极小解 $\boldsymbol{x}'^*_{\mathrm{min}}$, 如果其对应的目标函数值小于 $\boldsymbol{x}_{\mathrm{trial}}$ 的目标函数值, 则 $\boldsymbol{x}^*_{\mathrm{min}}$ 取代 $\boldsymbol{x}_{\mathrm{trial}}$ 并转 Step 5.6, 否则转 Step 5.7;

Step 5.6: 由式 (6) 计算出 $\boldsymbol{x}^*_{\mathrm{min}}$ 的支撑向量, 根据条件关系 (12) 和 (13) 更新树, 并转 Step 3;

Step 5.7: 由式 (6) 计算出 $\boldsymbol{x}_{\mathrm{trial}}$ 的支撑向量, 根据条件关系 (12) 和 (13) 更新树, 并转 Step 3.

Step 6: 设置 $g = g + 1$, 并转 Step 3.

注 2 常数 M 应设置得足够大, 本文设置为 80 000; Step 2、Step 5.6 和 Step 5.7 中树结构的更新过程, 以及 Step 5.2 中找出包含 $\boldsymbol{x}_{\mathrm{trail}}$ 个体的树叶子节点的方法参见文献 [7]; Step 4 中变异、交叉操作参见文献 [10].

2.4 复杂度分析

根据算法流程进行时间复杂度分析. Step 1、Step 2 和 Step 5 中构建支撑面的时间复杂度为 $O(N+1)$, Step 5.2 中找出包含 x_{trial} 的树叶子节点时间复杂度为 $O(\log_{N+1}^{T})$, 其中 T 为 n 叉树的节点数, 故整个算法的平均时间复杂度为 $O((N+1)(\log_{N+1}^{T}))$.

3 数值实验与分析

3.1 测试函数及参数设置

为了验证本文算法的性能, 这里选用基本差分进化算法 DE, Kaelo 等[15]提出的基于锦标赛机制的改进差分算法 (DERL) 和采用反射与收缩算子的改进差分算法 (DELB), Qin 等[16]提出的自适应差分进化算法 (SaDE) 以及 Mallipeddi 等[17]提出的具有系综变异策略和参数的差分进化算法 (EPSDE) 5 种算法来进行对比, 同时应用 6 个典型的具有代表性的全局优化问题来保证实验结果的客观性. 6 个问题包含了多模态问题及单模态问题, 表 1 列出了各测试函数的参数.

表 1　标准测试函数

函数名称	表 达 式	维数 N	取值范围	全局最小值(解)
Griewank	$f_1(\boldsymbol{x})=1+\frac{1}{4\,000}\sum_{i=1}^{N}x_i^2-\prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$10,30$	$(-600,600)^N$	$0(0,\cdots,0)$
Exponential	$f_2(\boldsymbol{x})=-\exp\left(-0.5\sum_{i=1}^{N}x_i^2\right)$	$10,30$	$(-1,1)^N$	$-1(0,\cdots,0)$
Ackley	$f_3(\boldsymbol{x})=-20\exp\left(-0.2\sqrt{N^{-1}\sum_{i=1}^{N}x_i^2}\right)-\exp\left(N^{-1}\sum_{i=1}^{N}\cos(2\pi x_i)\right)+20+e$	$10,30$	$(-30,30)^N$	$0(0,\cdots,0)$
Rastrigin	$f_4(\boldsymbol{x})=10N+\sum_{i=1}^{N}[x_i^2-10\cos(2\pi x_i)]$	$5,10$	$(-5.12,5.12)^N$	$0(0,\cdots,0)$
Schaffer	$f_5(\boldsymbol{x})=\sum_{i=1}^{N-1}(x_i^2+x_{i+1}^2)^{0.25}(\sin^2(50(x_i^2+x_{i+1}^2)^{0.1})+1)$	$2,5$	$(-100,100)^N$	$0(0,\cdots,0)$
Rosenbrock	$f_6(\boldsymbol{x})=\sum_{i=1}^{N-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$2,3$	$(-2,2)^N$	$0(1,\cdots,1)$

ACUP 算法参数设置: $F=0.5$, $C_R=0.5$; DE 算法设置参见文献[10]; DERL 和 DELB 算法设置参见文献[15]; SaDE 和 EPSDE 算法分别参见文献[16]和[17]. 所有算法种群规模除了 f_6 函数设置为 30 外, 其余函数均设置为 20, 且各测试函数均独立运行 100 次. 实验基于 Intel Core i5-2410 M 2.30 GHz, 4 GB, Windows 7, Visual Studio 2010 环境, 算法采用 C++ 语言编程实现.

3.2　计算代价和可靠性

通过目标函数评价次数(FE)和算法成功率(SR)两个指标来验证 ACUP 算法在计算代价和可靠性方面的优势, 见表 2. 其中 SR 为成功运行次数与总运行次数之比. 算法终止条件为 $|f(\boldsymbol{x}_{\rm best})-{\rm opti}|\leqslant 0.000\,01$, "opti" 为全局最小值.

对比表 2 中的数据可知: 因为 ACUP 算法利用下界信息安全排除了部分局部极小值区域, 从一定程度上避免了早熟收敛, 所以算法的可靠性最高, 平均成功率为 0.972, DE 算法次之, SaDE 算法最低; 由于 ACUP 算法又利用下界信息指导种群进化, 其计算代价最小, 平均目标函数评价次数为 7 021, 相对于 DE、DERL、DELB、SaDE 和 EPSDE 算法分别节省了 51.5%、25.8%、6.8%、16.2% 和 11.2%; DE 算法虽然可靠性较高, 但其计算代价太大; DELB 和 EPSDE 算法虽然计算代价稍高于 ACUP 算法, 但其可靠性较低; 而 ACUP 算法在保证算法可靠性的同时, 计算代价也得到了改善.

表 2　目标函数评价次数和成功率

Fun	N	DE		DERL		DELB		SaDE		EPSDE		ACUP	
		FE	SR	FE	SR	FE	SR	FE	SR	FE	SR	FE	SR
f_1	30	35 216	0.92	17 460	0.73	14 491	0.34	19 140	0.33	13 987	0.37	15 011	0.96
	10	29 870	0.74	17 337	0.43	12 311	0.70	10 235	0.60	15 157	0.63	13 390	0.91
f_2	30	12 720	1.00	9 117	0.98	6 851	0.94	9 756	1.00	5 618	1.00	6 268	1.00
	10	3 996	1.00	3 021	1.00	2 173	1.00	1 938	1.00	1 968	1.00	1 826	1.00
f_3	30	16 759	1.00	12 416	0.83	16 822	0.57	23 680	0.07	13 686	0.43	10 416	1.00
	10	5 897	1.00	4 499	0.97	4 739	0.97	5 413	1.00	5 035	1.00	4 028	1.00
f_4	10	29 524	0.79	19 069	0.65	15 905	0.75	8 108	0.91	14 375	0.90	16 009	0.93
	5	7 451	0.96	5 616	0.90	3 538	0.97	3 876	1.00	3 862	1.00	3 792	1.00
f_5	5	14 477	1.00	11 785	0.99	6 308	0.98	7 351	1.00	7 747	1.00	6 529	1.00
	2	6 535	0.96	4 462	0.97	2 358	0.97	3 303	1.00	3 571	1.00	2 434	0.98
f_6	3	8 911	0.80	6 919	0.64	3 606	0.89	4 792	0.90	5 480	0.86	3 369	0.88
	2	2 264	1.00	1 840	0.98	1 278	1.00	2 934	1.00	4 350	1.00	1 184	1.00
AVE		14 468	0.931	9 462	0.839	7 532	0.840	8 377	0.818	7 903	0.849	**7 021**	**0.972**

3.3　平均收敛速度

基于算法 100 次独立运行的平均结果, 对各测试函数选取一个维数, 绘制平均收敛速度曲线进行比较分析, 结果如图 2 所示. 为了图形的清晰直观, 对于最优值为 0 的函数, 纵坐标设置为函数值的对数, 且当取值等于 -5 时, 即为达到最优.

从图 2 可以看出: ACUP 算法在对 6 个测试函数问题进行优化求解时, 除了 f_2-30 维问题后期收敛速度稍逊于 SaDE 算法外, 其余问题收敛速度均优于其他 5 种算法; 在对 f_1-30 维及 f_4-10 维问题求解时, 其

余5种算法均陷入了局部最优解, 只有ACUP算法能够成功求解. 总体而言, 利用下界信息指导种群进化, 同时又借助支撑面的下降方向作局部增强的ACUP算法, 不仅不易出现早熟收敛现象, 而且收敛速度较快.

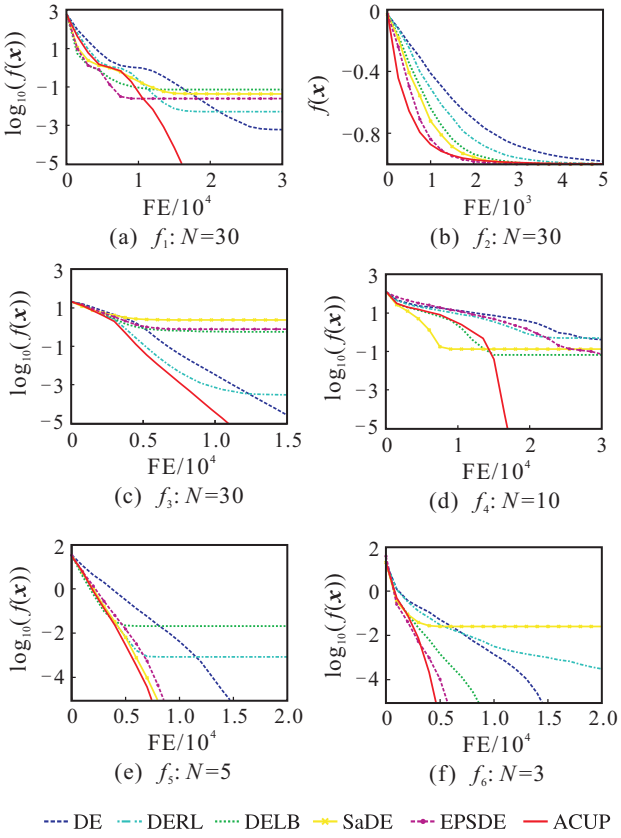

图 2 平均收敛速度曲线

4 结 论

在群体进化算法框架下, 结合抽象凸理论, 本文提出了一种基于抽象凸下界估计的群体全局优化算法, 实现了进化算法与抽象凸理论的协同优化框架, 不仅减少了算法的函数评价次数, 提高了可靠性, 而且加快了收敛速度, 同时又降低了计算复杂度. 实验数据表明, 所提出算法是一种有效的全局优化算法, 且提出的种群更新方法可以应用到其他群体算法中. 下一步的研究工作将结合局部搜索来进一步降低算法的复杂度.

参考文献(References)

[1] Avriel M. Nonlinear programming analysis and methods[M]. New Jersey: Prentice Hall, 1976.

[2] Nelder J A, Mead R A. A simplex method for function minimization[J]. Computer Journal, 1965, 7(7): 308-313.

[3] Walsh G R. Methods of optimization[M]. London: Wiley Press, 1975: 76-90.

[4] Tuy H. Convex analysis and global optimization[M]. Dordrecht: Kluwer Academic Publishers, 1998.

[5] Norkin V I, Pflug G C, Ruszczyński A. A branch and bound method for stochastic global optimization[J]. Mathematical Programming, 1998, 83(1/2/3): 425-450.

[6] Rubinov A M. Abstract convexity and global optimization[M]. Dordrecht: Kluwer Academic Publishers, 2000.

[7] Beliakov G. Geometry and combinatorics of the cutting angle method[J]. Optimization, 2003, 52(4): 379-394.

[8] Floudas C A, Gounaris C E. A review of recent advances in global optimization[J]. J of Global Optimization, 2009, 45(1): 3-38.

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

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

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

[12] Kennedy J. Particle swarm optimization[C]. Encyclopedia of Machine Learning. New York: Springer, 2010: 760-766.

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

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

[15] Kaelo P, Ali M M. A numerical study of some modified differential evolution algorithm[J]. European J of Optimization, 2006, 169(3): 1176-1184.

[16] Qin A K, Huang V L, Suganthan P N. Differential evolution algorithm with strategy adaptation for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2009, 13(2): 398-417.

[17] Mallipeddi R, Suganthan P N, Pan Q K, et al. Differential evolution algorithm with ensemble of parameters and mutation strategies[J]. Applied Soft Computing, 2011, 11(2): 1679-1696.

(责任编辑: 李君玲)