基于构造学习的差分进化算法求解部分可分优化问题

陈作汉[1,2*]，曹　洁[1,2]，赵付青[1]，张建林[1]

(1. 兰州理工大学计算机与通信学院　兰州　730050；2. 甘肃省制造业信息化工程技术研究中心　兰州　730050)

【摘要】复杂优化问题中决策变量之间的依赖性导致函数适应度地形中存在大量的局部最优解，传统进化算法求解此类问题相对困难。提出一种求解部分可分函数优化问题的构造学习差分进化算法 CLSHADE。该算法首先利用差分分组技术将复杂问题解耦划分为多个子问题，降低问题复杂程度；然后基于分组结构设计一种构造学习策略，以一定概率向构造的最优解学习以引导种群的搜索方向，提高算法搜索性能。在 CEC 2017 部分可分测试函数上的实验结果表明了 CLSHADE 的有效性。

关　键　词　构造学习；　差分进化；　差分分组；　部分可分问题

中图分类号　TP18　　**文献标志码**　A　　doi:10.12178/1001-0548.2022082

A Constructive Learning Differential Evolution Algorithm for Partially Separable Function Optimization Problems

CHEN Zuohan[1,2*], CAO Jie[1,2], ZHAO Fuqing[1], and ZHANG Jianlin[1]

(1. College of Computer and Communication, Lanzhou University of Technology　Lanzhou　730050;

2. Gansu Engineering Research Center of Manufacturing Informationization　Lanzhou　730050)

Abstract　The dependence among decision variables in complex optimization problems leads to the appearance of a large number of local optimal solutions in the fitness landscape of functions, which are difficult to be solved by classical evolutionary algorithms. In this paper, a constructive learning success-history based adaptive differential evolution (CLSHADE) algorithm is proposed to solve partially separable function optimization problems. Firstly, CLSHADE uses the differential grouping technology (DG) to reduce the complexity of a complex problem by dividing it into multiple sub-problems. Then, a constructive learning strategy, based on the grouping structure, is designed. It learns from the constructed optimal solution in a certain probability to guide the search direction and improve the search performance of the CLSHADE. The experimental results on the partially separable function of CEC 2017 demonstrate the effectiveness of the CLSHADE.

Key words　constructive learning strategy;　differential evolution;　differential grouping;　partially separable problems

现代工程实践和生产经营等领域存在大量的复杂优化问题，如流程制造车间过程控制、复杂工业环境生产调度、航空发动机控制及生物学中的基因网络等[1]。这些问题往往具有非线性、多模态、决策变量之间耦合程度高等特点，传统优化方法求解上述问题的表现并不理想。进化算法为解决此类复杂优化问题提供了新的思路[2-3]。

差分进化[4](differential evolution, DE) 算法由于原理简单、控制参数少及鲁棒性高等优点，得到了广泛关注。目前，针对 DE 算法的改进主要集中在 3 个方面。1) DE 参数的自适应策略研究，由于控制参数对算法性能有重要影响，围绕 DE 的关键参数如缩放因子 F、交叉概率 Cr 和种群规模 NP 等产生了大量的研究结果，如 SaDE[5]、JADE[6] 及 SHADE[7] 等。2) 从变异策略方面对 DE 算法进行改进，通过参与变异个体选择、差分扰动项、向最优解学习项等的变化产生了如 DE/rand/1、DE/best/1、DE/best/2、DE/rand/2、DE/current-to-best/1 等大量

收稿日期：2022－03－21；修回日期：2022－04－10

基金项目：国家重点研发计划 (2020YFB1713600)；国家自然科学基金 (62063021)

作者简介：陈作汉 (1979－)，男，博士生，主要从事智能优化算法和无线传感网络等方面的研究.

*通信作者：陈作汉，E-mail：chenzh@lut.edu.cn

有代表性的研究成果[8-9]。3) 结合其他算法如PSO[10]，CMA-ES[11]等，融入其他算法的优点从种群初始化、个体选择、进化过程等方面提升DE的性能。

尽管大量DE及其改进算法展现出了良好的性能，然而随着决策变量的增加及决策变量之间存在大量复杂耦合关系，算法的求解性能出现一定程度的下降，甚至部分复杂优化问题的最优解依然没有找到。另一方面，耦合决策变量之间的交互关系没有进行解耦分析以降低问题复杂程度，决策变量间的结构信息也没有得到充分的挖掘和利用，而这些信息往往对降低问题复杂度和指导种群搜索方向有较大帮助[12-13]。

基于以上分析，本文针对决策变量部分可分离复杂优化函数问题，基于“分而治之”思想采用差分分组策略对耦合决策变量进行解耦分组，形成多个没有依赖关系的简单子问题，然后设计一种构造学习策略，利用多个优秀个体的潜在有用信息，以每个子问题的优秀维度组合构造当前种群的最优解，引导种群向构造的最优解学习，提高算法的搜索性能，为解决具有复杂多变量耦合特征的部分可分优化问题提供基于差分进化算法的解决方法。

1 相关基础算法

本文引入差分分组 (differential grouping, DG2)[14]策略，基于算法SHADE(success-history based adaptive DE)[7]进行改进，以下分别进行介绍。

1.1 SHADE

SHADE算法包含初始化、变异、交叉和选择等步骤。首先在问题的可行域中随机产生NP个D维向量构成初始种群：

$$\boldsymbol{X}_i=(x_1,x_2,\cdots x_D)\quad i=1,2,\cdots,\mathrm{NP} \tag{1}$$

变异操作是DE算法的核心操作，目前已有许多经典的变异策略，SHADE算法采用DE/current-to-pbest/1变异策略：

$$\boldsymbol{V}_i^t=\boldsymbol{X}_i^t+F(\boldsymbol{X}_{\mathrm{pbest}}^t-\boldsymbol{X}_i^t)+F(\boldsymbol{X}_{r1}^t-\boldsymbol{X}_{r2}^t) \tag{2}$$

式中，t为当前进化代数；r_1、r_2随机从[1,NP]中选取且与i互不相等；F为缩放因子；$\boldsymbol{X}_{\mathrm{pbest}}^t$随机从$\mathrm{NP}\times p(p\in[0.11,0.2])$个当前种群的最优解中选取。SHADE采用一个外部存档存储保存部分较差的个体以增加种群的多样性。式(2)表明DE/current-to-pbest/1变异策略包含了差分扰动和向优秀解学习两部分，$F(\boldsymbol{X}_{r1}^t-\boldsymbol{X}_{r2}^t)$为扰动项，且$\boldsymbol{X}_{r2}^t$从外部存档和当前种群的合集中随机选取。$F(\boldsymbol{X}_{\mathrm{pbest}}^t-\boldsymbol{X}_i^t)$为向当前优秀解学习项。如果变异向量$V_i^t$超出问题的可行域，则执行以下操作：

$$\boldsymbol{V}_{i,j}^t=\begin{cases}(x_{j,\min}+x_{i,j}^t)/2 & \boldsymbol{V}_{i,j}^t<x_{j,\min}\\(x_{j,\max}+x_{i,j}^t)/2 & \boldsymbol{V}_{i,j}^t>x_{j,\max}\end{cases} \tag{3}$$

交叉操作将变异向量和父代向量进行交叉，形成实验向量：

$$\boldsymbol{U}_{i,j}^t=\begin{cases}\boldsymbol{V}_{i,j}^t & j=j_{\mathrm{rand}}\ \text{or}\ \mathrm{rand}(0,1)\leqslant \mathrm{Cr}\\\boldsymbol{X}_{i,j}^t & \text{otherwise}\end{cases} \tag{4}$$

式中，Cr是交叉概率；j_{rand}从$[1,D]$之间随机选取。然后从目标向量和实验向量中选择适应度较小 (以最小化问题为例) 的个体进入下一代种群，如下所示：

$$\boldsymbol{U}_{i,j}^t=\begin{cases}\boldsymbol{U}^t & f(\boldsymbol{U}^t)\leqslant f(\boldsymbol{X}_i^t)\\\boldsymbol{X}_i^t & \text{otherwise}\end{cases} \tag{5}$$

参数控制机制对算法的搜索性能起着至关重要的作用，SHADE采用一种基于成功历史的自适应策略，将进化过程中成功的H个缩放因子F和交叉概率Cr保存在M_{Cr}和M_F中以供选择。参数F服从柯西分布$C(\mu_F,0.1)$，μ_F的初值为0.5，更新如下：

$$\mu_F=(1-c)\mu_F+c\,\mathrm{mean}_L(S_F) \tag{6}$$

式中，S_F为子代成功取代父代时参数F的集合，$\mathrm{mean}_L(S_F)$为Lehmer均值。类似地，交叉概率Cr服从正态分布$N(\mu_{\mathrm{Cr}},0.1)$，μ_{Cr}的初值为0.5，更新如下：

$$\mu_{\mathrm{Cr}}=(1-c)\mu_{\mathrm{Cr}}+c\,\mathrm{mean}_A(S_{\mathrm{Cr}}) \tag{7}$$

式中，S_{Cr}为子代成功取代父代时参数Cr的集合；$\mathrm{mean}_A(S_{\mathrm{Cr}})$为算术平均。

1.2 差分分组策略

基于“分而治之 (divide-and-conquer)”的思想，分组是进化算法解决复杂函数优化问题的一种重要策略，该策略将复杂问题划分为多个相对简单的子问题。有效的分组策略可以解耦多个存在交互关系的决策变量，最小化不同子问题之间的依赖关系[15]。

函数$f(\boldsymbol{X}):\mathbb{R}^n\to\mathbb{R}$是部分可分离问题，当且仅当：

$$\begin{aligned}&\arg\min_{(x_1,x_2,\cdots,x_n)} f(x_1,x_2,\cdots x_n)=\\&(\arg\min_{\boldsymbol{X}_1} f(x_1,\cdots),\cdots,\arg\min_{\boldsymbol{X}_m} f(,\cdots x_n))\end{aligned} \tag{8}$$

式中，$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$是$n$维决策向量；$\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_m$是向量$\boldsymbol{X}$的不相交子向量，$2\leqslant m\leqslant n$。

函数$f(\boldsymbol{X})$满足如下条件则称之为部分加性可分离：

$$f(\boldsymbol{X})=\sum_{i=1}^{n} f_i(x_i) \quad n>1 \tag{9}$$

差分分组方法检测决策变量之间的交互关系，如果存在交互则将其分在一组，否则被分在不同的组，如果一个决策变量与其他所有变量都不存在交互，则该变量为独立变量。分组方法定义如下。

$f(\boldsymbol{X})$是部分加性可分离函数，$\forall a,b_1,b_2,x_p,x_q\in\mathbb{R}$，$a,b_1\neq b_2$，$\delta\in\mathbb{R}$，$\delta\neq 0$，如果以下式子成立，则$x_p$和$x_q$存在交互关系，是不可分决策变量。

$$\Delta_{\delta,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_1}\neq\Delta_{\delta,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_2} \tag{10}$$

$$\Delta_{\delta,x_p}[f](\boldsymbol{x})=f(\cdots,x_p+\delta,\cdots)-f(\cdots,x_p,\cdots) \tag{11}$$

否则，x_p和x_q为可分离决策变量。

2　构造学习差分进化算法 CLSHADE

CLSHADE 的核心步骤主要包含 3 部分：分组、构造学习策略及 SHADE。首先对复杂问题通过差分分组检测决策变量间的耦合关系，将原始问题分解为多个相互独立的分组。然后基于母体算法 SHADE 通过构造学习策略提高其搜索能力。算法总体框架如图 1 所示。

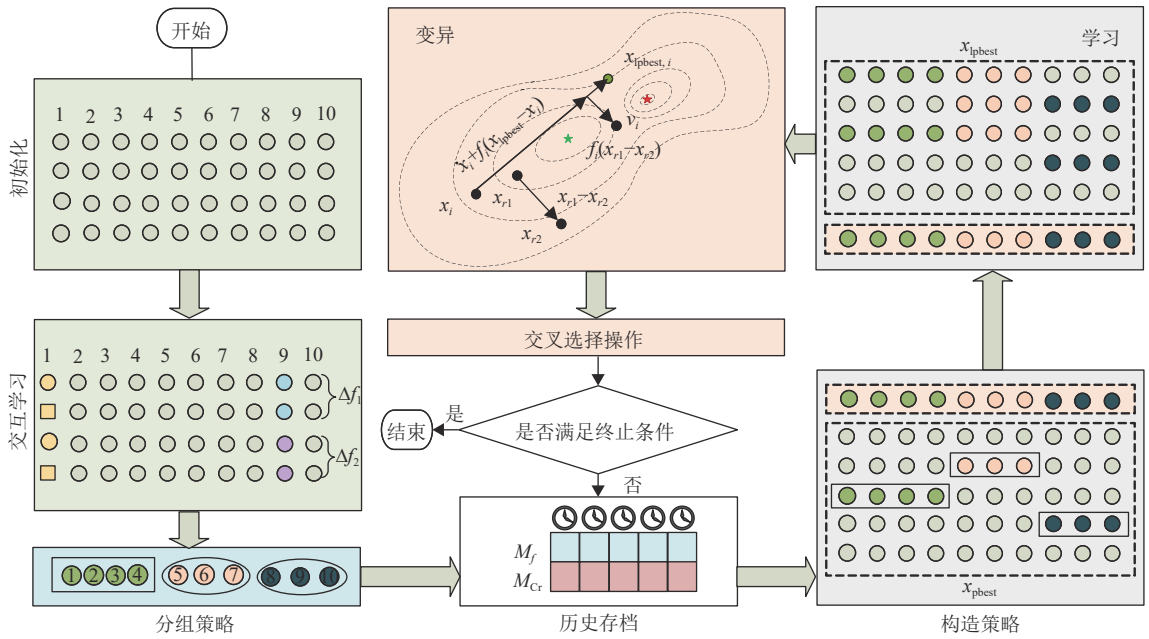


图 1　算法总体框架

2.1　问题分组

问题分组策略检查每个决策变量与其他决策变量之间是否存在交互关系，循环上述过程直到完成所有变量之间交互关系的判断，具体如算法 1 所示。其中$f(\boldsymbol{X})$为目标函数，$x^{\min}$和$x^{\max}$为决策变量上下界，$\mathrm{ones}(1,D)$函数用于生成D维全 1 向量。

为判断决策向量的第i维和第j维是否存在交互，初始化两个决策向量$\boldsymbol{x}_1$和$\boldsymbol{x}_2$，$\boldsymbol{x}_1$初始化为决策变量下界，$\boldsymbol{x}_2$除第j维初始化为上界外，其他维度和$\boldsymbol{x}_1$一样初始化为下界，然后根据式 (10) 和 (11) 计算Δ_1和Δ_2，根据既定阈值ε判断交互关系($\varepsilon=10^{-3}$)，最后输出决策变量分组结果。

算法 1　分组算法 DG

输入：$f(\boldsymbol{X})$

输出：分组结果groups

初始化：$S\leftarrow\{1,2,\cdots,D\}$,sep $\leftarrow\{\}$,groups $\leftarrow\{\}$

For $i\in S$ do

　group $\leftarrow\{i\}$

　For $j\in S\wedge i\neq j$ do

　　$\boldsymbol{x}_1\leftarrow x^{\min}\times\mathrm{ones}(1,D),\boldsymbol{x}_2\leftarrow\boldsymbol{x}_1$

　　$\boldsymbol{x}_2(i)\leftarrow x^{\max},\Delta_1=f(\boldsymbol{x}_1-\boldsymbol{x}_2)$

　　$\boldsymbol{x}_1(j)\leftarrow 0,\boldsymbol{x}_2(j)\leftarrow 0,\Delta_2=f(\boldsymbol{x}_1-\boldsymbol{x}_2)$

　　//初始化并根据式 (10)～式 (11) 计算Δ

　　If $|\Delta_1-\Delta_2|>\varepsilon$ then

　　group $\leftarrow$ group $\cup\{j\}$

　　//如果相关则放入一组

　　End

　End

$S \leftarrow S - \text{group}$ //相关则从S中移除
If length(group) = 1 Then
seps ← seps ∪ group //保存独立变量
Else
groups ← groups ∪ group
//在groups中加入新识别的分组
End
End
groups ← groups ∪ {seps}//输出结果

2.2 构造学习

变异策略是 DE 算法的核心操作，DE/best/1、DE/best/2 以及 SHADE 采用的 DE/current-to-best/1 都向当前种群的优秀解$\boldsymbol{X}^t_{\text{pbest}}$学习，利用$\boldsymbol{X}^t_{\text{pbest}}$中潜在的有用信息引导种群的搜索。然而，$\boldsymbol{X}^t_{\text{pbest}}$有可能存在部分维度优秀，部分维度一般的情况。一个简单的例子，假设给定一个最小化优化函数$f(x) = (x_1 - x_2)^2 + (x_3 - x_4)^2$，$x \in [0,1]^4$，基于交互关系决策变量显然可以分为两组$\{x_1, x_2\}$，$\{x_3, x_4\}$。对于 3 个$\boldsymbol{X}^t_{\text{pbest}}$：$(0.8, 0.4, 0.6, 0.4)$，$(0.6, 0.4, 0.8, 0.4)$和$(0.6, 0.4, 0.6, 0.4)$，前两个解都在部分分组维度上优秀，而第三个解在全部分组维度上优秀，最终最优解为$(0.6, 0.4, 0.6, 0.4)$。

得益于分组结构，构造一个$\boldsymbol{X}^t_{\text{cpbest}}$，组合多个个体不同分组维度的优秀信息，具体如下：

$$\boldsymbol{X}^t_{\text{cpbest}} = (\boldsymbol{X}^t_{\text{pbest},1}, \boldsymbol{X}^t_{\text{pbest},2}, \cdots, \boldsymbol{X}^t_{\text{pbest},m}) \tag{12}$$

式中，$\boldsymbol{X}^t_{\text{pbest},i}$为第$i$个分组的优秀解。新的变异策略结合 DE/current-to-best/1 策略和构造学习策略(constructive learning strategy, CLS)，如下所示：

$$\boldsymbol{V}^t_i = \boldsymbol{X}^t_i + F(\boldsymbol{X}^t_{\text{lpbest}} - \boldsymbol{X}^t_i) + F(\boldsymbol{X}^t_{r1} - \boldsymbol{X}^t_{r2}) \tag{13}$$

式中，$\boldsymbol{X}^t_{\text{lpbest}}$以概率$Pc_i$向构造的$\boldsymbol{X}^t_{\text{cpbest}}$学习，如：

$$\boldsymbol{X}^t_{\text{lpbest}} = \begin{cases} \boldsymbol{X}^t_{\text{cpbest}} & \text{rand} < Pc_i \\ \boldsymbol{X}^t_{\text{pbest}} & \text{rand} \geqslant Pc_i \end{cases} \tag{14}$$

$$Pc_i = 0.05 + \varphi \times \frac{\left(\exp\left(\frac{10(i-1)}{\text{ps}-1}\right)\right)}{(\exp(10) - 1)} \tag{15}$$

式 (15) 为学习概率Pc_i的定义，其中，φ为学习因子；ps为$\boldsymbol{X}^t_{\text{pbest}}$的数量；rand是$[0,1]$之间的随机数。可以看出$Pc_i$单调递增，是考虑到随着进化过程的推进，$\boldsymbol{X}^t_{\text{cpbest}}$逐步向问题最优解区域靠近，因此在进化后期比进化前期赋予相对较高的学习概率。考虑到学习因子φ对算法性能的影响，因此在本文实验部分和其他重要参数一起进行正交实验分析。构造学习策略具体流程如算法 2 所示。

算法 2　构造学习策略 CLS
输入：$f(\boldsymbol{X})$, groups = $\{\text{group}_1, \text{group}_2, \cdots, \text{group}_m\}$
输出：$\boldsymbol{X}_{\text{lpbest}}$
For $i = 1 : \text{ps}$ do //构造$\boldsymbol{X}_{\text{cpbest,i}}$过程
For $j = 1 : m$ do
$\tau = \text{groups}(j)$ //第j个分组
$\text{ind}^j_1 = \lceil \text{rand}(0,1) \times \text{ps} \rceil$
$\text{ind}^j_2 = \lceil \text{rand}(0,1) \times \text{ps} \rceil$ //向上取整
If $f(\boldsymbol{X}_{\text{pbest}}(\text{ind}^j_1)) \leqslant f(\boldsymbol{X}_{\text{pbest}}(\text{ind}^j_2))$
$\boldsymbol{X}^\tau_{\text{cpbest,i}} = \boldsymbol{X}_{\text{pbest}}(\text{ind}^j_1)$
Else
$\boldsymbol{X}^\tau_{\text{cpbest,i}} = \boldsymbol{X}_{\text{pbest}}(\text{ind}^j_2)$
End
End
End //基于分组构造最优解
For $i = 1 : \text{ps}$ do //$\boldsymbol{X}^t_{\text{lpbest},i}$的生成
For $j = 1 : m$ do
$\tau = \text{groups}(j)$
If rand < Pc_i
$\boldsymbol{X}^\tau_{\text{lpbest},i} = \boldsymbol{X}^\tau_{\text{cpbest},i}$
Else
$\boldsymbol{X}^\tau_{\text{lpbest},i} = \boldsymbol{X}^\tau_{\text{pbest},i}$
End
End
End //向构造的最优解学习

2.3 CLSHADE 算法

CLSHADE 首先采用算法 1 对问题分组，然后在 SHADE 算法中使用算法 2 设计的构造策略迭代寻优，具体流程如算法 3 所示。

算法 3　CLSHADE
输入：$f(\boldsymbol{X})$，维数D
输出：最优解$\boldsymbol{X}_{\text{best}}$
初始化：$\text{NP} = 18D$; $\boldsymbol{X}^0 = (\boldsymbol{X}^0_1, \boldsymbol{X}^0_2, \cdots, \boldsymbol{X}^0_{\text{NP}})$; $t = 0$
$H = 5$; $M_{\text{Cr},i} = 0.5$, $M_{F,i} = 0.5$;
groups = $\text{DG}(f(\boldsymbol{X}), \boldsymbol{X}^0)$ //采用算法 1 进行分组
Repeat
$S_{\text{CR}} = \phi, S_F = \phi$
$\boldsymbol{X}_{\text{lpbest}} = \text{CLS}(f(\boldsymbol{X}), \text{groups})$
//采用算法 2 实施构造学习
For $i = 1 : \text{NP}$ do
$k = \text{rand}[1, H]$ //从历史存档中随机选择

$\mathrm{CR}_i = \mathrm{rand}n_i(M_{\mathrm{Cr},k}, 0.1)$

$F_i = \mathrm{rand}c_i(M_{F,k}, 0.1)$

$p_i = \mathrm{rand}[p_{\min}, 0.2]$

采用式 (13) 进行变异操作

采用式 (4) 进行交叉操作

采用式 (5) 进行选择操作

End

更新$M_{\mathrm{Cr},k}, M_{F,k}$

$t++$ //迭代次数递增

Until 达到算法终止条件

3　实验和分析

3.1　实验环境及参数设置

本文实验环境为：Windows 10 操作系统；Intel (R) Core (TM) i7-6700 CPU @ 3.40 GHz 处理器；8 GB 内存；基于 MATLAB 实现。比较算法的控制参数按照相应文献的建议进行设置。选取 CEC 2017[16] 标准测试集的变量部分可分函数并在 10D、30D 上采用相同的评价准则进行实验。经过实验发现 CEC 2017 标准测试函数f_1～f_{20}为部分变量部分可分函数，因此选用f_1～f_{20}作为测试函数。

CLSHADE 算法有 3 个关键的参数：变异因子M_F、交叉概率M_{Cr}和学习因子φ，其初始取值的设定对算法的性能有重要影响，本文采用实验设计方法 DOE(design of experiment)[17] 分析其对算法性能的影响并最终确定相应的初始取值。表 1 为每个参数的水平值设置。

表 1　参数水平设置

参数	水平			
	1	2	3	4
M_F	0.3	0.5	0.7	0.9
M_{Cr}	0.3	0.5	0.7	0.9
φ	0.35	0.45	0.55	0.65

表 1 参数的所有可能组合数为$4\times4\times4=64$，每个参数组合独立运行 10 次以确保实验具有统计学意义。对实验结果进行多元方差分析 (multivariate analysis of variance, MANOVA)，结果如表 2 所示。

从表 2 可以看出，参数φ对应的 F-ratio 最大，即参数φ对 CLSHADE 的平均性能影响最大。由于算法两个参数之间的p值大于 0.05，所以 CLSHADE 中两个参数之间没有交互作用。参数主效应图如图 2 所示，因此选取参数取值组合：$M_F=0.3$, $M_{\mathrm{Cr}}=0.5$, $\varphi=0.45$。其他 CLSHADE 算法中涉及的所有其他非关键参数都与 SHADE 保持一致。

表 2　参数方差分析结果

Source	Sum of squares	Degrees of freedom	Mean Square	F-ratio	p-value
M_F	0.410	3	0.137	0.43	0.730 3
M_{Cr}	0.591	3	0.197	0.63	0.604 5
φ	1.549	3	0.516	**1.64**	**0.203 3**
M_F*M_{Cr}	2.780	9	0.309	0.98	0.477 1
$M_F*\varphi$	2.259	9	0.251	0.8	0.621 6
$M_{\mathrm{Cr}}*\varphi$	0.782	9	0.868	0.28	0.976 0
Error	8.498	27	0.315		
Total	16.869	63			

表 3　Rastrigin's 函数的构造学习数据

$\boldsymbol{X}_{\mathrm{pbest}}$	x_1	x_2	x_3	x_4	x_5	x_6	$F(\boldsymbol{x})$
1	−43.67	26.11	−28.75	−79.55	−38.51	71.29	535.73
2	−22.06	−4.18	49.64	−13.93	19.72	0.24	538.07
3	−32.94	18.91	−28.04	−14.42	22.07	83.43	540.03
4	37.80	70.86	−60.65	−26.23	49.67	12.95	541.39
5	−68.07	−39.78	−27.09	−42.74	25.82	74.39	541.48
6	**23.36**	**83.11**	−8.96	−97.31	17.70	31.17	541.90
7	10.48	16.73	38.69	38.44	13.00	−37.07	542.02
8	−21.81	18.78	27.74	−42.74	−3.60	28.81	545.09
9	−87.73	60.57	16.15	41.29	−35.96	−8.70	545.45
10	−40.63	98.65	72.85	31.97	34.68	−24.68	546.14
11	−15.94	−33.65	24.52	−62.09	−24.69	−22.56	547.46
12	24.66	−46.47	**20.06**	**-64.49**	**44.71**	**14.16**	547.82
$\boldsymbol{X}_{\mathrm{lpbest}}$	23.36	83.11	20.06	−64.49	44.71	14.16	**512.49**

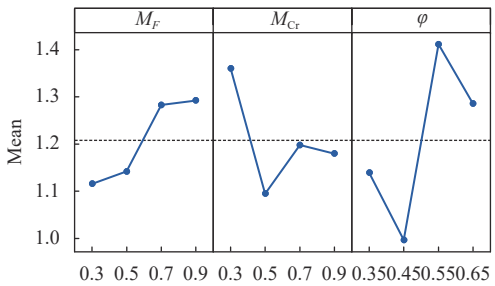


图 2　参数主效应图

3.2　算法运行实例分析

为通过具体函数观察 CLSHADE 的运行机制，以求解 CEC 2017 测试集的$F_5(\boldsymbol{x})$(shifted and rotated rastrigin's function) 函数为例进行说明。从图 3 所示 Rastrigin's 函数的适应度地形图可以看出，该函数具有多模及多个局部最优等复杂特性。

CEC 2017 对其进行旋转平移定义如下：

$$F(\boldsymbol{x}) = f(\boldsymbol{M}(\boldsymbol{x}-\boldsymbol{o})) + 500 \tag{16}$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{17}$$

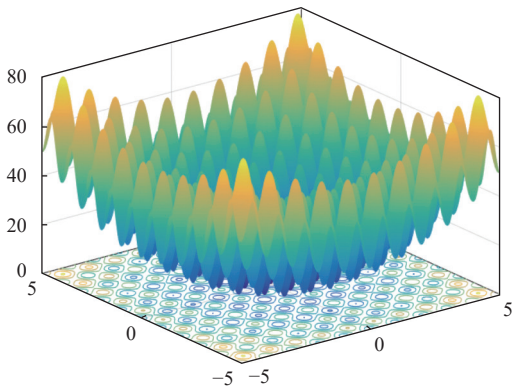


图 3 Rastrigin's 函数的适应度地形图

以$D=6$为例，其旋转矩阵$\boldsymbol{M}$和平移向量$\boldsymbol{o}$分别为：

$$\boldsymbol{M} = \begin{pmatrix} 0.09 & 0.61 & 0.00 & 0.00 & 0.00 & 0.00 \\ -0.54 & 0.68 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.48 & 0.00 & 0.00 & 0.11 \\ 0.00 & 0.00 & 0.00 & 0.53 & -0.52 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.43 & 0.85 & 0.00 \\ 0.00 & 0.00 & -0.93 & 0.00 & 0.00 & 0.00 \end{pmatrix} \tag{18}$$

$$\boldsymbol{o} = (\ -17.41 \quad 56.17 \quad -31.76 \quad -56.76 \quad 16.67 \quad 79.12\ ) \tag{19}$$

采用算法 1 进行对 Rastrigin's 函数进行分组，共分为 3 个组，分别是 $\text{group}_1 = \{x_1, x_2\}$, $\text{group}_2 = \{x_3, x_6\}$, $\text{group}_3 = \{x_4, x_5\}$。采用构造学习策略后$F(\boldsymbol{x})$的值如表 3 所示，显然得到了一个更有潜力的优秀解，从而加快了算法的收敛速度，并提高了解的精度。

3.3 与其他算法对比

为验证 CLSHADE 算法的性能，选取 SHADE[7]、AMECoDEs[18]、HS-ES[19] 和 EBLSHADE[20] 作为比较算法。所有实验在每个测试问题上独立运行 51 次，计算均值和标准差 (Mean_{std})，实验结果如表 4 (10 维) 和表 5(30 维) 所示。

表 4 列出了 10 维问题的实验结果，可以看出所有算法在f_1～f_4、f_9等 5 个函数上都找到了相同的最优解，在其余 15 个函数上 CLSHADE 有 7 个函数的结果优于其他算法，3 个函数和部分其他算法找到相同的最优解，在其余函数如f_5、f_{13}和f_{19}上，CLSHADE 也与求解性能最优算法的结果没有显著性差异。综上所述，CLSHADE 在 10 维问题上的总体表现优于其他对比算法。

表 4 10 维问题实验结果

Func	$\text{SHADEMean}_{\text{std}}$	$\text{AMECoDEsMean}_{\text{std}}$	$\text{HS-ESMean}_{\text{std}}$	$\text{EBLSHADEMean}_{\text{std}}$	$\text{CLSHADEMean}_{\text{std}}$
1	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
2	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
3	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
4	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
5	2.05E+00$_{8.06E-01}$	7.14E+00$_{2.85E+00}$	**5.66E−01**$_{7.24E-01}$	1.97E+00$_{7.83E-01}$	7.70E−01$_{7.70E-01}$
6	**0.00E+00**$_{0.00E+00}$	4.60E−06$_{3.42E-06}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
7	1.20E+01$_{5.69E-01}$	1.65E+01$_{2.77E+00}$	1.13E+01$_{7.82E-01}$	1.19E+01$_{4.66E-01}$	**1.06E+01**$_{1.06E+01}$
8	2.19E+00$_{8.45E-01}$	6.52E+00$_{2.74E+00}$	6.83E−01$_{7.57E-01}$	2.24E+00$_{7.92E-01}$	**1.68E−01**$_{1.68E-01}$
9	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
10	1.26E+01$_{2.38E+01}$	1.58E+02$_{1.24E+02}$	1.13E+02$_{1.49E+02}$	8.47E+00$_{7.94E+00}$	**4.67E−01**$_{4.67E-01}$
11	**0.00E+00**$_{0.00E+00}$	1.03E+00$_{9.58E-01}$	1.81E+00$_{8.55E+00}$	1.20E−03$_{8.56E-03}$	**0.00E+00**$_{0.00E+00}$
12	1.45E+01$_{3.98E+01}$	1.25E+01$_{3.67E+01}$	2.88E+01$_{5.94E+01}$	**2.45E−01**$_{1.80E-01}$	9.52E+00$_{9.52E+00}$
13	**3.08E+00**$_{2.31E+00}$	4.31E+00$_{1.98E+00}$	3.32E+00$_{2.46E+00}$	3.26E+00$_{2.29E+00}$	3.13E+00$_{3.13E+00}$
14	1.93E−02$_{1.29E-01}$	1.75E+00$_{1.38E+00}$	5.11E+00$_{1.84E+01}$	3.08E−03$_{1.64E-02}$	**9.35E−04**$_{9.35E-04}$
15	**5.52E−03**$_{1.91E-02}$	1.20E−01$_{3.00E-01}$	6.13E−01$_{6.26E-01}$	6.67E−02$_{1.60E-01}$	2.53E−02$_{2.53E-02}$
16	1.25E−01$_{1.65E-01}$	6.04E−01$_{1.53E+00}$	5.10E+00$_{3.04E+01}$	1.66E−01$_{2.11E-01}$	**7.51E−02**$_{7.51E-02}$
17	8.89E−03$_{1.43E-02}$	1.27E+00$_{8.30E-01}$	1.39E+01$_{1.15E+01}$	1.63E−02$_{4.96E-02}$	**3.20E−03**$_{3.20E-03}$
18	7.61E−02$_{1.66E-01}$	6.05E−02$_{2.36E-01}$	4.48E−01$_{1.85E-01}$	7.53E−02$_{1.47E-01}$	**5.40E−02**$_{5.40E-02}$
19	6.03E−05$_{3.30E-04}$	3.59E−02$_{4.41E-02}$	8.79E−01$_{4.49E+00}$	**1.02E−08**$_{7.31E-08}$	4.85E−08$_{4.85E-08}$
20	**0.00E+00**$_{0.00E+00}$	8.68E−02$_{1.94E-01}$	1.31E+01$_{9.60E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$

表 5 为 30 维问题的实验结果，所有算法在函数f_1～f_3、f_9上取得相同的最优解，在f_6上 CLSHADE 和 HS-ES、EBLSHADE 都获得最优解，在剩余的 15 个函数中，CLSHADE 在 10 个函数上优于其他算法，3 个函数上与表现最优算法接近。总体来说，CLSHADE 在 30 维问题上表现优秀。

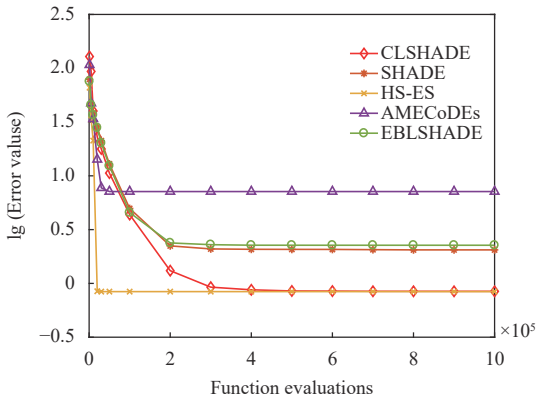
表 5　30 维问题实验结果

Func	SHADEMean$_{std}$	AMECoDEsMean$_{std}$	HS-ESMean$_{std}$	EBLSHADEMean$_{std}$	CLSHADEMean$_{std}$
1	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
2	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
3	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
4	5.87E+01$_{7.78E-01}$	5.89E+01$_{1.32E+00}$	**5.29E+00**$_{1.20E+01}$	5.86E+01$_{0.00E+00}$	5.86E+01$_{0.00E+00}$
5	5.81E+00$_{1.16E+00}$	2.80E+01$_{1.07E+01}$	8.29E+00$_{3.18E+00}$	5.48E+00$_{1.23E+00}$	**2.84E+00**$_{9.01E-01}$
6	6.71E−10$_{4.79E-09}$	1.07E−08$_{3.72E-08}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
7	3.71E+01$_{9.51E-01}$	5.63E+01$_{9.29E+00}$	4.07E+01$_{6.57E+00}$	3.63E+01$_{9.61E-01}$	**3.47E+01**$_{6.11E-01}$
8	6.37E+00$_{1.17E+00}$	2.60E+01$_{1.13E+01}$	7.96E+00$_{2.29E+00}$	6.07E+00$_{1.44E+00}$	**3.19E+00**$_{8.40E-01}$
9	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$	**0.00E+00**$_{0.00E+00}$
10	1.21E+03$_{1.85E+02}$	1.60E+03$_{5.33E+02}$	9.90E+02$_{3.88E+02}$	1.07E+03$_{1.56E+02}$	**5.99E+02**$_{1.11E+02}$
11	1.67E+01$_{2.41E+01}$	2.16E+01$_{2.41E+01}$	1.59E+01$_{2.42E+01}$	1.93E+01$_{2.62E+01}$	**1.45E+01**$_{2.22E+01}$
12	9.68E+02$_{3.57E+02}$	9.47E+02$_{3.45E+02}$	**3.46E+01**$_{8.72E+01}$	8.16E+02$_{2.98E+02}$	9.32E+02$_{2.99E+02}$
13	1.33E+01$_{5.19E+00}$	2.11E+01$_{8.54E+00}$	1.53E+01$_{4.30E+00}$	**1.15E+01**$_{6.46E+00}$	1.32E+01$_{4.84E+00}$
14	5.88E+00$_{8.83E+00}$	2.35E+01$_{8.05E+00}$	1.43E+01$_{9.95E+00}$	8.92E+00$_{9.69E+00}$	**3.97E+00**$_{7.28E+00}$
15	1.97E+00$_{1.48E+00}$	6.31E+00$_{3.04E+00}$	3.60E+00$_{3.28E+00}$	1.46E+00$_{1.09E+00}$	**1.24E+00**$_{9.96E-01}$
16	2.06E+01$_{4.03E+00}$	4.75E+02$_{2.07E+02}$	2.38E+02$_{1.66E+02}$	**1.41E+01**$_{2.98E+00}$	2.19E+01$_{4.71E+00}$
17	2.37E+01$_{6.26E+00}$	3.06E+01$_{4.55E+01}$	3.53E+01$_{6.28E+01}$	2.19E+01$_{5.57E+00}$	**1.48E+01**$_{3.04E+00}$
18	1.97E+01$_{4.74E+00}$	2.59E+01$_{6.39E+00}$	1.93E+01$_{4.96E+00}$	1.91E+01$_{5.27E+00}$	**1.85E+01**$_{6.38E+00}$
19	**1.65E+00**$_{7.67E-01}$	4.68E+00$_{1.53E+00}$	4.06E+00$_{2.49E+00}$	1.80E+00$_{9.36E-01}$	1.99E+00$_{8.61E-01}$
20	2.17E+01$_{5.88E+00}$	2.76E+01$_{1.98E+01}$	1.67E+02$_{7.53E+01}$	1.81E+01$_{6.52E+00}$	**1.76E+01**$_{4.30E+00}$

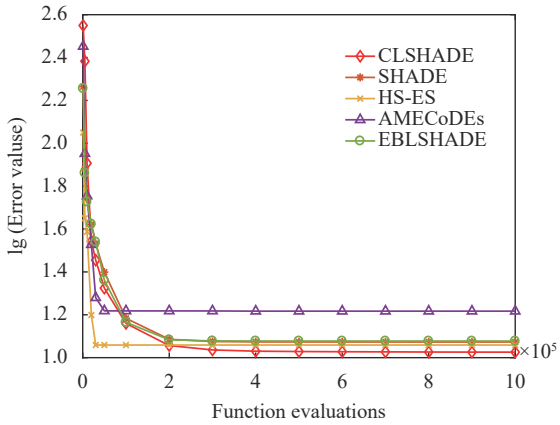
3.4　分析与讨论

为进一步比较和分析所有算法的性能，本文以 10 维问题为例，通过收敛图观察算法的搜索过程 (收敛性)，通过盒图考察最优解的分布情况 (稳定性)，最后用 Wilcoxon 检验分析算法的统计学特性，结果如图 4～图 5 及表 6 所示。

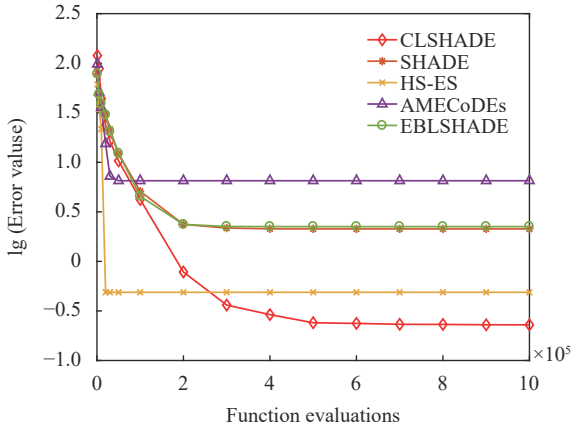
图 4 列出了部分函数的收敛图，可以看出 CLSHADE 算法与其他算法在进化前期与其他算法收敛速度接近。但是随着进化迭代次数的增加，构造的$\boldsymbol{X}^t_{\text{cpbest}}$更接近最优解所在的区域，其引导搜索的作用更加明显，因此算法执行后期收敛速度加快并且解的精度也有明显的提升，如图中函数f_8、f_{10}、f_{16}及f_{19}等所示。



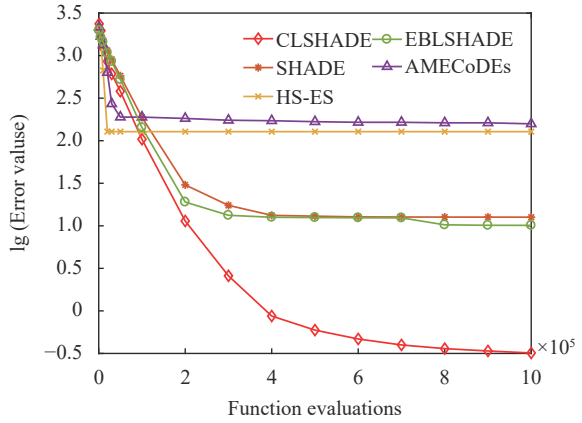
a. f_5 函数收敛曲线



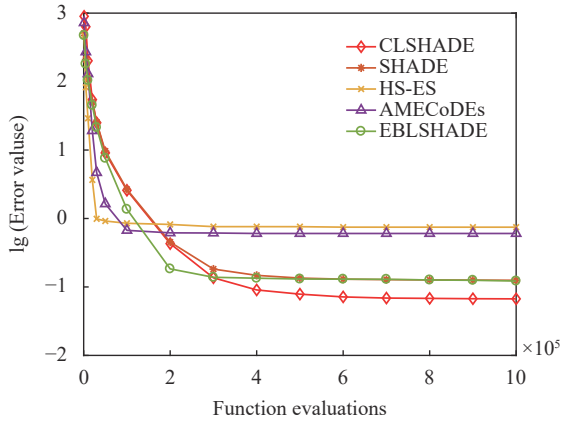
b. f_7 函数收敛曲线



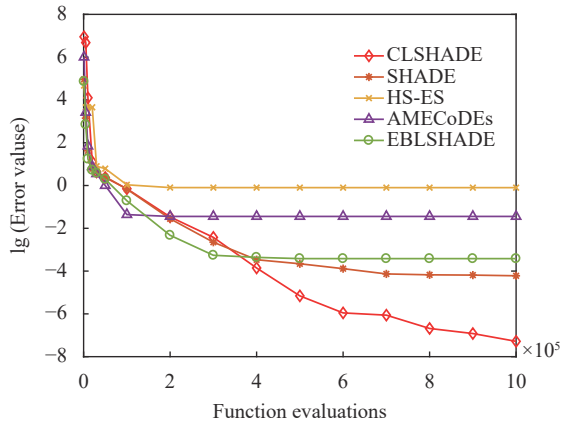
c. f_8 函数收敛曲线



d. f_{10} 函数收敛曲线



e. f_{16} 函数收敛曲线

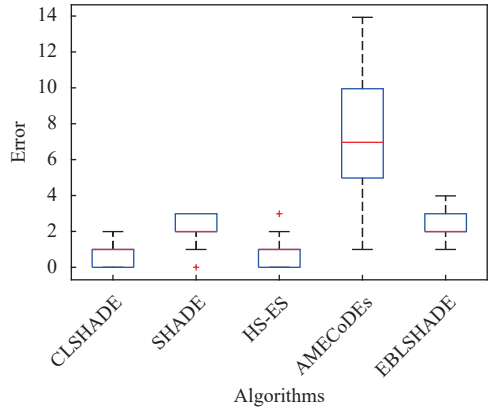


f. f_{19} 函数收敛曲线

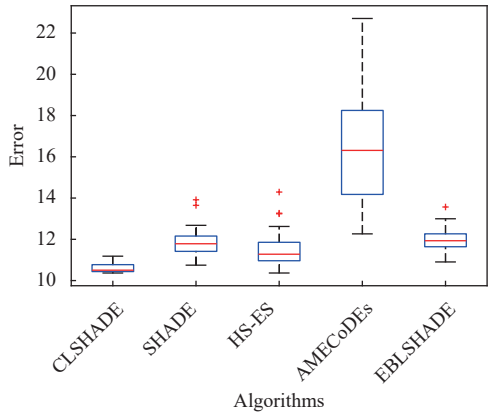
图 4　10 维问题部分函数收敛曲线

图 5 反映了与图 4 对应函数各算法最优解的分布情况，可以看出 CLSHADE 算法在函数 f_7、f_8和 f_{10}上具有良好的分布特性，表现优秀；在函数 f_5、f_{16}和 f_{19}上与其他优秀算法表现相当，解的分布都比较集中，具有良好的稳定性。

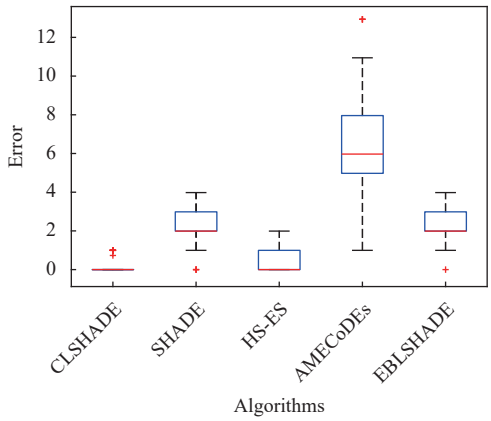
表 6 为 CLSHADE 与其他算法在显著性水平为 0.05 和 0.1 下的 Wilcoxon 非参数检验结果，可以看出在所有 10 维问题上 CLSHADE 与其他算法存在显著差异，都优于其他对比算法。在 30 维问题上，CLSHADE 显著优于 SHADE 和 AMECoDEs，当显著水平为 0.1 时优于 HS-ES 算法，其他情况下虽然不存在显著性差异，但从R–值明显大于R+也可以得出 CLSHADE 优于其他算法。因此，从统计学的角度来说，CLSHADE 总体表现优于其他对比算法。



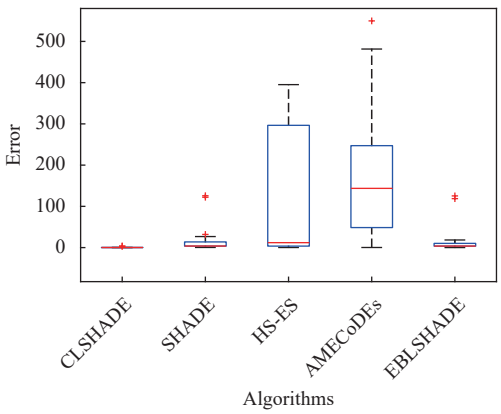
a. f_5 函数解的分布



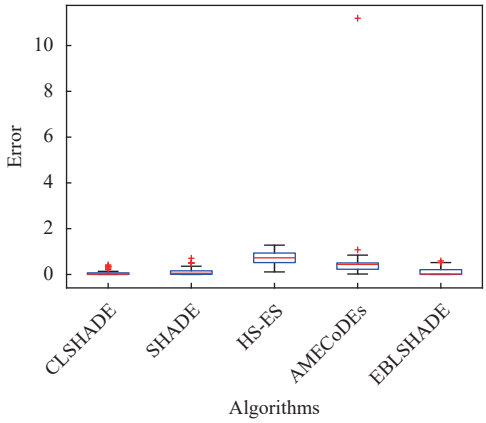
b. f_7 函数解的分布



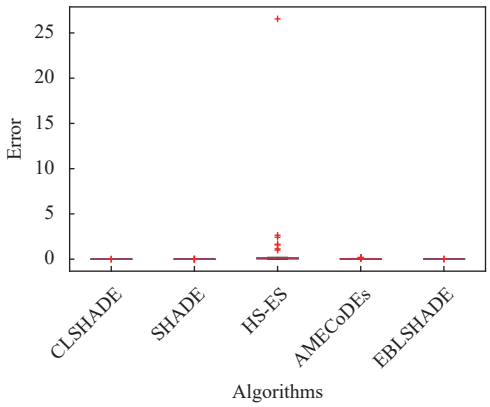
c. f_8 函数解的分布



d. f_{10} 函数解的分布



e. f_{16} 函数解的分布



f. f_{19} 函数解的分布

图 5　10 维问题部分函数解的分布盒图

表 6　Wilcoxon 检验结果

维	CLSHADE vs	R+	R−	Z	*p*-value	$\alpha = 0.1$	$\alpha = 0.05$
10	SHADE	17.00	74.00	−1.992	0.046	**Yes**	**Yes**
	AMECoDEs	0.00	120.00	−3.408	0.001	**Yes**	**Yes**
	HS-ES	1.00	104.00	−3.233	0.001	**Yes**	**Yes**
	EBLSHADE	26.00	79.00	−1.664	0.096	**Yes**	**Yes**
30	SHADE	11.00	125.00	−2.948	0.003	**Yes**	**Yes**
	AMECoDEs	0.00	136.00	−3.516	0.000	**Yes**	**Yes**
	HS-ES	26.00	94.00	−1.931	0.053	**Yes**	No
	EBLSHADE	32.00	73.00	−1.287	0.198	No	No

4　结 束 语

本文提出了一种基于构造学习的差分进化算法求解变量部分可分函数优化问题，采用差分分组技术揭示决策变量的底层交互结构和形成子成分并进行解耦分组；通过构造学习策略利用问题潜在的结构知识构造最优解并向其学习提升算法的寻优性能。在 CEC 2017 部分可分测试函数上的实验结果表明，本文算法在解的精度、收敛性和稳定性等方面均表现突出。

参 考 文 献

[1] MA X, LI X, ZHANG Q, et al. A survey on cooperative co-evolutionary algorithms[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(3): 421-441.

[2] LI J Y, ZHAN Z H, WANG H, et al. Data-Driven evolutionary algorithm with perturbation-based ensemble surrogates[J]. IEEE Transactions on Cybernetics, 2021, 51(8): 3925-3937.

[3] LI S, GONG W, YAN X, et al. Parameter estimation of photovoltaic models with memetic adaptive differential evolution[J]. Solar Energy, 2019, 190: 465-474.

[4] 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.

[5] 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.

[6] ZHANG J, SANDERSON A C. JADE: Adaptive differential evolution with optional external archive[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(5): 945-958.

[7] TANABE R, FUKUNAGA A. Success-History based parameter adaptation for differential evolution[C]//IEEE Congress on Evolutionary Computation (CEC 2013). [S.l.]: IEEE, 2013: 71-78.

[8] DAS S, MULLICK S S, SUGANTHAN P N. Recent advances in differential evolution: An updated survey[J]. Swarm and Evolutionary Computation. 2016, 27: 1-30.

[9] 陈作汉, 曹洁, 赵付青, 等. 基于搜索偏好知识的复杂多模差分进化算法[J]. 电子科技大学学报, 2020, 49(6): 875-882.
CHEN Z H, CAO J, ZHAO F Q, et al. Complex multimodal differential evolution algorithm based on search preference knowledge[J]. Journal of the University of Electronic Science and Technology of China, 2020, 49(6): 875-882.

[10] WANG S H, LI Y Z, YANG H Y. Self-Adaptive mutation differential evolution algorithm based on particle swarm optimization[J]. Applied Soft Computing. 2019, 81(8): 1-22.

[11] OPARA K, JAROSŁA W. Comparison of mutation strategies in differential evolution: A probabilistic

perspective[J]. Swarm and Evolutionary Computation, 2018, 39: 53-69.

[12] CHENG R, OMIDVAR M N, GANDOMI A H, et al. Solving incremental optimization problems via cooperative coevolution[J]. IEEE Transactions on Evolutionary Computation, 2019, 23(5): 762-775.

[13] QIANG Y, CHEN W N, GU T, et al. Segment-Based predominant learning swarm optimizer for large-scale optimization[J]. IEEE Transactions on Cybernetics, 2016, 47(9): 1-15.

[14] OMIDVAR M N, MING Y, YI M, et al. DG2: A faster and more accurate differential grouping for large-scale black-box optimization[J]. IEEE Transactions on Evolutionary Computation, 2019, 21(6): 929-942.

[15] OMIDVAR M N, LI X, MEI Y, et al. Cooperative co-evolution with differential grouping for large scale optimization[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 378-393.

[16] AWAD N H, ALI M Z, SUGANTHAN P N, et al. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization[R]. Singapore: Nanyang Techndogcal University, 2016.

[17] ANGELA M D, DANIEL T V. Design and analysis of experiments[M]. Cham: Springer, 2017.

[18] CUI L, LI G, ZHU Z, et al. Adaptive multiple-elites-guided composite differential evolution algorithm with a shift mechanism[J]. Information Sciences, 2018, 422(1): 122-143.

[19] GENG Z, SHI Y. Hybrid sampling evolution strategy for solving single objective bound constrained problems[C]// IEEE Congress on Evolutionary Computation (CEC 2018). [S.l.]: IEEE, 2018: 1-7.

[20] MOHAMED A W, HADI A A, JAMBI K M. Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization[J]. Swarm and Evolutionary Computation, 2019, 50(11): 1-14.

编 辑 蒋 晓