•信息工程•

DOI:10.15961/j.jsuese.201800273

基于共享学习策略的微分进化算法

段美军[1], 杨红雨[1,2], 刘　洪[2*], 陈俊逸[1], 刘　宇[2]

(1.四川大学 视觉合成图形图像技术国防重点学科实验室, 四川 成都 610065; 2.四川大学 计算机学院, 四川 成都 610065)

摘　要:针对传统微分进化算法易发生早熟收敛问题,提出基于共享学习策略的微分进化算法(SLDE),引入共享个体和共享学习因子。共享个体覆盖整个种群,较优个体可引导算法朝希望方向进化,较差个体则能维持种群的多样性,向共享个体学习可避免丢失个体信息,实现整个种群间的信息交换,有助于算法跳出局部最优解,提高算法的局部开采和全局勘探能力。同时,算法充分利用个体的进化信息,根据个体适应值到最优适应值的距离自适应地调整共享学习因子,以弥补随机个体对进化带来的随机性和盲目性,增强算法的搜索能力。采用22个不同特性的Benchmark测试函数对算法进行性能测试,与7种改进DE算法进行性能对比,实验结果表明,SLDE具有较强的跳出局部最优解能力,能显著减少进化代数,大幅地提高算法的收敛精度、收敛速度和稳定性,SLDE的全局优化性能整体上远优于其他改进DE算法。

关键词:微分进化; 共享学习; 共享个体; 共享学习因子; 自适应

中图分类号:TP18　　**文献标志码**:A　　**文章编号**:2096-3246(2019)01-0205-08

Differential Evolution Algorithm Based on Sharing Learning

DUAN Meijun[1], *YANG Hongyu*[1,2], *LIU Hong*[2*], *CHEN Junyi*[1], *LIU Yu*[2]

(1.National Key Lab. of Fundamental Sci. on Synthetic Vision, Sichuan Univ., Chengdu 610065, China;
2.College of Computer Sci., Sichuan Univ., Chengdu 610065, China)

Abstract: In order to alleviate premature convergence in traditional differential evolution algorithms, a differential evolution algorithm based on sharing learning strategy (SLDE) was proposed and the concepts of sharing-individual (SI) and sharing learning factor were introduced in SLDE. The sharing-individual covers the whole population, while the superior individuals guide the promising searching direction, the inferior individuals maintain the population diversity in the evolution process. By learning from the sharing-individual, information exchange was achieved among the whole population to avoid missing information of individuals, which helps the algorithm jump over the trap of local optimal solution and improve the local and global exploration capability. Meanwhile, the evolutionary information of individuals was made full of use in SLDE, and the sharing learning factor was self-adaptively adjusted according to the distance of fitness value of individual and the optimal fitness value, in order to alleviate the randomness and blindness from the random individuals and enhance the searching ability. A total of 22 Benchmark test functions with different properties were used for performance test comparison with seven state-of-the-art DE variants. The experimental results showed that SLDE has strong ability to escape from local optima, significantly reduce the evolutionary generations and greatly improve the convergence precision, convergence speed and stability. The overall global optimization performance of SLDE is much better than other improved DE algorithms.

Key words: differential evolution; sharing learning; sharing-individual; sharing learning factor; self-adaptiveness

微分进化算法(differential evolution, DE)是由Storn和Price针对实参优化问题提出的一种基于群体的随机优化方法[1]。DE算法原理简单,可调参数少,收敛速度快,鲁棒性好。目前,DE广泛用于神经网络参数训练、模式识别、机器人路径规划等科研和工程领域[2–7]。

收稿日期:2018 – 03 – 13

基金项目:国家重大科学仪器设备开发专项资助(2013YQ49087905);国家自然科学基金委员会与中国民用航空局联合资助项目(U1833115)

作者简介:段美军(1982—)女,博士生. 研究方向:智能计算;空中交通管理. E-mail: dmj23@163.com

* 通信联系人 E-mail: liuhong@scu.edu.cn

网络出版时间:2019 – 01 – 16 11 : 31 : 07　　**网络出版地址**: http://kns.cnki.net/kcms/detail/51.1773.TB.20190115.1417.004.html

传统的DE算法与其他进化算法一样容易陷入局部最优，存在早熟收敛问题。因此，许多学者分别针对控制参数和变异模式提出了一系列改进的DE算法。

1）着重于控制参数自适应的算法：Liu等[8]利用模糊逻辑控制调节缩放因子（F）和交叉概率（CR），但是如何确定模糊隶属函数比较困难；Brest等[9]基于DE/rand/1提出一种自适应DE算法，在进化过程中按指定的阈值动态地调整F和CR，但其阈值固定，缺乏进化过程的指导；Noman等[10]根据父代种群的平均适应值和子代个体的适应值动态调整F和CR；Tanabe等[11]提出一种基于历史优秀参数而实现控制参数自适应的技术。

2）侧重改进变异模式的算法：Zhang等[12]通过改进DE/current-to-best/1模式，提出DE/current-to-pbest/1变异模式，分别按正态分布和柯西分布随机生成F和CR，且采用保持优秀参数的机制，有较优的优化性能；Qin[13]、Wang[14]、Mallipeddi[15]、Wu[16]等基于一些变异模式的优势，通过组合多种变异策略改善算法的寻优能力。但是，这些算法计算量较大，复杂度高。Gong等[17]引入一种基于个体适应值等级修复交叉概率的策略；刘昊等[18]基于择优学习的思想进行变异，算法的复杂度较小；Wang等[19]基于DE/rand/1提出一种改进的自适应算法（IMMSADE），控制参数是否调整由固定的阈值决定。

针对传统微分进化算法易发生早熟收敛问题，为进一步提高算法的寻优性能，作者提出了一种基于共享学习策略的微分进化算法（differential evolution based on sharing learning, SLDE）。选取22个不同特性的Benchmark测试函数对算法进行性能测试，与7种改进DE算法进行性能对比，实验结果表明，SLDE算法能显著减少进化代数，有效地打破局部优化积累，大幅提高收敛精度和稳定性。

1 微分进化算法

DE基于群体进化，标准DE算法包括变异、交叉以及选择操作，对于D维优化问题：

$$\begin{cases}\min\ f(x_1,x_2,\cdots,x_D)\\ \text{s.t. } L_j\le x_j\le U_j,\ j=1,2,\cdots,D\end{cases}\tag{1}$$

式中，L_j与U_j分别为第j维变量的下边界和上边界。

与其他进化算法变异算子采用定义的概率分布函数不同，DE算法从当前种群中选取个体进行差分运算，并乘以缩放因子实现个体变异。常用的5类典型的变异策略如下：

1）DE/best/1

$$\boldsymbol{V}_i^{t+1}=\boldsymbol{X}_{\rm b}^t+F(\boldsymbol{X}_{r1}^t-\boldsymbol{X}_{r2}^t)\tag{2}$$

2）DE/best/2

$$\boldsymbol{V}_i^{t+1}=\boldsymbol{X}_{\rm b}^t+F(\boldsymbol{X}_{r1}^t-\boldsymbol{X}_{r2}^t)+F(\boldsymbol{X}_{r3}^t-\boldsymbol{X}_{r4}^t)\tag{3}$$

3）DE/rand/1

$$\boldsymbol{V}_i^{t+1}=\boldsymbol{X}_{r1}^t+F(\boldsymbol{X}_{r2}^t-\boldsymbol{X}_{r3}^t)\tag{4}$$

4）DE/rand/2

$$\boldsymbol{V}_i^{t+1}=\boldsymbol{X}_{r1}^t+F(\boldsymbol{X}_{r2}^t-\boldsymbol{X}_{r3}^t)+F(\boldsymbol{X}_{r4}^t-\boldsymbol{X}_{r5}^t)\tag{5}$$

5）DE/current-to-best/1

$$\boldsymbol{V}_i^{t+1}=\boldsymbol{X}_i^t+F(\boldsymbol{X}_{\rm b}^t-\boldsymbol{X}_i^t)+F(\boldsymbol{X}_{r1}^t-\boldsymbol{X}_{r2}^t)\tag{6}$$

式中：$i=\{1,2,\cdots,N_p\}$，N_p为种群大小；$t=\{1,2,\cdots,T\}$为进化代数；$\boldsymbol{X}_{\rm b}^t$为当代种群中最优个体；$\boldsymbol{X}_i^t$为对应的父代个体；$\boldsymbol{X}_{r1}^t\sim\boldsymbol{X}_{r5}^t$为不同于$\boldsymbol{X}_i^t$的随机选择个体，且$r_1\ne r_2\ne r_3\ne r_4\ne r_5$；$F\in[0,2]$为缩放因子。

通过对变异向量进行交叉操作生成实验向量$\boldsymbol{U}_i^{t+1}=[U_{i,1}^{t+1},U_{i,2}^{t+1},\cdots,U_{i,D}^{t+1}]$，可增加种群的多样性以扩大搜索区域。常用的二项式交叉操作为：

$$U_{i,j}^{t+1}=\begin{cases}V_{i,j}^{t+1},\ \text{rand}(0,1)\le CR\ \text{ or }\ j=j_{\rm rand};\\ X_{i,j}^t,\ \text{others}\end{cases}\tag{7}$$

式中：交叉概率$CR\in[0,1]$；若CR较大，则会加快算法的收敛；若CR较小，则算法的鲁棒性会更好；$j_{\rm rand}$为$[1,D]$之间的随机整数。

DE算法采用贪婪策略进行选择操作，根据目标向量$\boldsymbol{X}_i^t$和实验向量$\boldsymbol{U}_i^{t+1}$的函数值选择最优个体，如式（8）所示：

$$\boldsymbol{X}_i^{t+1}=\begin{cases}\boldsymbol{X}_i^{t+1},f(\boldsymbol{X}_i^{t+1})<f(\boldsymbol{X}_i^t);\\ \boldsymbol{X}_i^t,\text{others}\end{cases}\tag{8}$$

2 SLDE算法

2.1 共享学习变异策略

典型变异策略中，DE/best/1和DE/best/2利用最优个体的信息而具有较强的局部开发能力，收敛速度较快，易早熟收敛；策略DE/rand/1和DE/rand/2仅基于随机个体的信息进行变异，具有较强的全局搜索能力，但缺乏最优个体信息的指导，收敛速度较慢；DE/current-to-best/1基于父代个体和最优个体进行变异，寻优精度较高，易陷入局部最优。为此，作者提出共享学习变异策略以提高算法的全局搜索和局部开发能力。

共享学习变异策略引入共享个体，通过最优个体与共享个体的差分学习各个个体之间的信息，并通过个体的适应值与最优适应值的距离自适应地调整共享学习因子。

$$\boldsymbol{V}_i^{t+1}=F(\boldsymbol{X}_{r1}^t-\boldsymbol{X}_{r2}^t)+\omega_i^t\cdot(\boldsymbol{X}_{\rm b}^t-\overline{\boldsymbol{X}}^t)\tag{9}$$

式中，$\boldsymbol{X}_{r1}^t$和$\boldsymbol{X}_{r2}^t$为当代种群中不同于父代个体的随机

个体，$\overline{\boldsymbol{X}}^t$为共享个体，$\omega_i^t$为共享学习因子。$\overline{\boldsymbol{X}}^t$、$\omega_i^t$定义如下：

$$\overline{\boldsymbol{X}}^t = \frac{1}{N_p}\left(\sum_{i=1}^{N_p}\boldsymbol{X}_i^t\right) \tag{10}$$

$$\omega_i^t = (f_{\max}^t - f(\boldsymbol{X}_i^t))/(f_{\max}^t - f_{\min}^t) \tag{11}$$

式(11)中，$f_{\max}^t$为种群最优适应值，$f(\boldsymbol{X}_i^t)$为父代个体$\boldsymbol{X}_i^t$的适应值，$f_{\min}^t$为种群最差适应值。

作者提出的共享学习变异策略具有以下特性：

1)共享个体涵盖所有个体的信息，较优个体可引导算法朝希望方向进化，较差个体则能维持种群的多样性。通过向共享个体学习可实现整个种群之间信息交换，避免丢失个体信息，提高算法的全局搜索与局部开发能力。

2)根据个体的适应值自适应地调整共享学习因子，共享学习因子控制最优个体和共享个体的差分对个体变异的作用，弥补了随机个体本身的随机性对进化带来的随机性和盲目性，加快算法的收敛速度。

2.2 SLDE算法

SLDE算法的主要步骤如下：

Step1：初始化种群，设置算法的初始参数：种群大小N_p、进化代数t、问题维数D、最大进化代数T、算法收敛精度ε、缩放因子F及交叉概率CR。

Step2：分别按式(10)、(11)更新共享个体及共享学习因子。

Step3：满足结束条件，终止算法并输出结果。

Step4：按式(9)的变异策略进行变异。

Step5：按式(7)、(8)进行交叉和选择操作。

Step6：令$t = t+1$，转到Step2继续执行。

3 算法性能分析

3.1 Benchmark测试函数

为测试算法的性能，选取22个不同特性的Benchmark测试函数进行性能测试，如表1所示。

3.2 与改进的DE比较

将SLDE算法与aDE[10]、SHADE[11]、SaDE[13]、CoDE[14]、EPSDE[15]、Rcr-JADE[17]和PLDE[18]算法进行比较。参数设置：f_{12}收敛精度为1E−3，f_{20}和f_{21}收敛精度为1E−1，其余函数的收敛精度为1E−8，种群大小为100，独立实验次数为30，30维问题和100维问题的最大迭代次数分别为1 500和3 000。SLDE算法中$F=0.2$，$CR=0.8$，其他算法控制参数的具体设置请参见相应的参考文献。

对算法的性能测试包括两个方面：

1)算法的收敛精度，如表2、3所示。

从表2、3可得：1)对于单峰函数f_1～f_4、f_6～f_8和f_{10}，SLDE均能寻优到全局最优解；在f_5和f_9上，SLDE优势明显。2)对于阶跃函数f_{11}和噪声函数f_{12}，SLDE不劣于其他算法。3)对于多峰函数f_{13}～f_{18}和f_{22}，SLDE能寻优到全局最优解；在f_{19}上，SLDE优于EPSDE，且不劣于其他算法；在f_{20}上，SLDE劣于其他算法。在f_{21}上，D=30时，SLDE劣于其他算法；D=100时，SLDE为最优。4)相同的种群规模下，随着函数问题维数和最大迭代次数的增加，SLDE算法仍维持高的收敛精度，而其他算法的收敛精度则明显下降。

同时，为进一步评估算法整体寻优性能，本文对算法的优化结果进行非参数检验。表4为Wilcoxon检验结果(显著水平分别为0.05和0.10)。表5为Friedman和Kruskal_Wallis检验结果。

表4中，R^+为SLDE优于对比算法检验秩的和，R^-为对比算法优于SLDE检验秩的和，R^+越大，SLDE算法更优。由表4可知，无论是高维还是低维问题，Wilcoxon检验所求得的R^+均大于对比算法，SLDE整体上均显著优于其他改进算法。

由表5可知，SLDE由Friedman和Kruskal_Wallis检验所得的平均秩均为最小，表明SLDE整体寻优性能最好，其次是Rcr-JADE(D=30)和SHADE(D=100)。

综合上述实验结果可以发现：SLDE以最优个体为基，由共享个体引导个体朝希望方向进化的策略是有效的，能较好地平衡算法的局部搜索和全局开发能力，从而提高算法的寻优精度。

2)算法的收敛速度和稳定性，如表6、7所示。

表6、7中，*MEGN*为达到指定收敛精度的平均进化代数，*SR*为实验成功率。分析表6、7中算法的收敛速度和稳定性可知，无论是低维问题还是高维问题，除f_{12}和f_{19}～f_{21}外，SLDE均以最少的迭代次数达到指定精度且实验成功率均为100，稳定性最好，收敛速度最快。

图1给出部分函数的进化曲线。

观察图1中两类典型的进化曲线可知：如图1(a)、(b)、(e)所示，SLDE能寻优到全局最优值，其他比较算法未陷入“进化停滞”，但其平均最优解的精度远差于SLDE；SLDE能在较少的进化代数内寻优到全局最优解，而SHADE[11]、SaDE[13]、CoDE[14]、EPSDE[15]、Rcr-JADE[17]和PLDE[18]则在不同的进化代数陷入“进化停滞”，如图1(c)、(d)、(f)所示。SLDE在进化中无明显的“进化停滞”，证明算法通过共享个体学习个体间的信息可帮助算法跳出局部最优解，提高算法的全局搜索能力。

表 1 Benchmark测试函数

Tab. 1 Benchmark test functions

函数	变量范围	最小值	属性
$f_1(x)=\sum_{i=1}^{D}x_i^2$	$[-100,100]^D$	0	单峰
$f_2(x)=\sum_{i=1}^{D}(10^6)^{\frac{i-1}{N-1}}x_i^2$	$[-100,100]^D$	0	单峰
$f_3(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i}x_j)^2$	$[-100,100]^D$	0	单峰
$f_4(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$	0	单峰
$f_5(x)=\max\{\lvert x_i\rvert,\ 1\le i\le D\}$	$[-100,100]^D$	0	单峰
$f_6(x)=\sum_{i=1}^{D}ix_i{}^2$	$[-10,10]^D$	0	单峰
$f_7(x)=-\exp(-0.5\sum_{i=1}^{D}x_i^2)$	$[-1,1]^D$	−1	单峰
$f_8(x)=10^6x_1^2+\sum_{i=2}^{D}x_2^2$	$[-100,100]^D$	0	单峰
$f_9(x)=\sum_{i=1}^{D}x_i^2+(\sum_{i=1}^{D}0.5ix_i)^2+(\sum_{i=1}^{D}0.5ix_i)^4$	$[-5,10]^D$	0	单峰
$f_{10}(x)=x_1^2+10^6\sum_{i=2}^{D}x_i^2$	$[-100,100]^D$	0	单峰
$f_{11}(x)=\sum_{i=1}^{D}(\lvert x_i+0.5\rvert)^2$	$[-100,100]^D$	0	阶跃函数
$f_{12}(x)=\sum_{i=1}^{D}ix_i^4+\mathrm{rand}[0,1)$	$[-1.28,1.28]^D$	0	噪声函数
$f_{13}(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	$[-5.12,5.12]^D$	0	多峰
$f_{14}(x)=\sum_{i=1}^{D}x_i^2/4\,000-\prod_{i=1}^{D}\cos(x_i/\sqrt{i})+1$	$[-600,600]^D$	0	多峰
$f_{15}(x)=\sum_{i=1}^{D}\left(0.5+\frac{\sin^2(\sqrt{x_i^2+x_{i+1}^2})-0.5}{(1+0.001(x_i^2+x_{i+1}^2))^2}\right)\ x_{D+1}=x_1$	$[-0.5,0.5]^D$	0	多峰
$f_{16}(x)=1-\cos(2\pi\sqrt{\sum_{i=1}^{D}x_i^2})+0.1\sqrt{\sum_{i=1}^{D}x_i^2}$	$[-100,100]^D$	0	多峰
$f_{17}(x)=20+e-20\exp(-0.2\sqrt{\sum_{i=1}^{D}x_i^2/D})-\exp(\sum_{i=1}^{D}\cos(2\pi x_i)/D)$	$[-32,32]^D$	0	多峰
$f_{18}(x)=\sum_{i=1}^{D}(x_i^2+x_{i+1}^2)^{0.25}(\sin(50(x_i^2+x_{i+1}^2)^{0.1})+1)$	$[-100,100]^D$	0	多峰
$f_{19}=418.982\,9D-\sum_{i=1}^{D}g(z_i),\ z_i=x_i+4.209\,687\,462\,275\,03\mathrm{E}+002,$ $g(z_i)=\begin{cases}z_i\sin(\lvert z_i\rvert^{1/2}),\lvert z_i\rvert<500;\\(500-\mathrm{mod}(z_i,500))\sin(\sqrt{\lvert 500-\mathrm{mod}(\lvert z_i\rvert,500)\rvert})-\frac{(z_i-500)^2}{1\,000D},\ z_i>500;\\(\mathrm{mod}(z_i,500)-500)\sin(\sqrt{\lvert\mathrm{mod}(\lvert z_i\rvert,500)-500\rvert})-\frac{(z_i-500)^2}{1\,000D},\ z_i<-500\end{cases}$	$[-100,100]^D$	0	多峰
$f_{20}(x)=\lvert(\sum_{i=1}^{D}x_i^2-D\rvert^{1/4}+(0.5\sum_{i=1}^{D}x_i^2+\sum_{i=1}^{D}x_i)/D+0.5$	$[-100,100]^D$	0	多峰
$f_{21}(x)=\lvert(\sum_{i=1}^{D}x_i^2)^2-(\sum_{i=1}^{D}x_i)^2\rvert^{1/2}+(0.5\sum_{i=1}^{D}x_i^2+\sum_{i=1}^{D}x_i)/D+0.5$	$[-100,100]^D$	0	多峰
$f_{22}(x)=\sum_{i=1}^{D}(\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i+0.5))])-D\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k\cdot 0.5)],a=0.5,b=3,k_{\max}=20$	$[-0.5,0.5]^D$	0	多峰

表 2　SLDE与改进算法的优化结果（*D*=30）

Tab. 2　Optimization results of SLDE and improved algorithms with 30 dimensions

函数	平均最优解							
	SLDE	aDE	SHADE	SaDE	CoDE	EPSDE	Rcr-JADE	PLDE
f_1	0.00E+00*	2.67E–21	2.31E–73	2.35E–37	9.21E–14	1.06E–07	6.40E–79	1.20E–54
f_2	0.00E+00*	3.95E–18	6.72E–69	8.61E–34	7.70E–11	7.43E–04	1.53E–74	1.02E–50
f_3	0.00E+00*	8.45E+01	5.28E–21	1.87E–02	1.67E–04	1.70E+04	2.64E–22	5.38E+00
f_4	0.00E+00*	3.25E–13	1.34E–34	3.68E–21	3.53E–08	1.26E–04	3.29E–38	2.02E–30
f_5	1.67E–92*	3.02E–02	3.30E–18	1.27E–03	2.58E–02	2.19E+00	8.46E–18	8.10E–07
f_6	0.00E+00*	3.95E–22	3.57E–74	3.69E–38	1.13E–14	5.55E–08	2.06E–80	1.19E–55
f_7	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*
f_8	0.00E+00*	2.28E–15	3.60E–68	4.40E–31	1.06E–07	7.41E–02	1.02E–72	1.74E–48
f_9	1.60E–202*	3.28E+00	7.18E–16	2.99E–04	1.71E–05	3.39E+02	9.19E–17	2.91E+00
f_{10}	0.00E+00*	1.70E–15	6.91E–66	1.97E–31	5.75E–08	1.07E–01	5.58E–72	1.80E–48
f_{11}	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*
f_{12}	1.25E+03*	1.25E+03*	1.25E+03*	1.25E+03*	1.25E+03*	1.25E+03*	1.25E+03*	1.25E+03*
f_{13}	0.00E+00*	1.55E+01	1.01E–14	1.34E–13	7.19E+00	3.25E+01	0.00E+00*	2.60E+01
f_{14}	0.00E+00*	0.00E+00*	0.00E+00*	2.47E–04	2.17E–11	8.01E–04	1.23E–03	1.81E–03
f_{15}	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*	0.00E+00*	1.93E–11	0.00E+00*	0.00E+00*
f_{16}	0.00E+00*	3.57E–22	3.19E–74	2.89E–38	2.65E–14	8.53E–09	1.95E–80	1.29E–55
f_{17}	0.00E+00*	1.47E–11	4.26E–15	3.55E–15	8.34E–08	6.24E–05	5.21E–15	5.33E–15
f_{18}	0.00E+00*	6.25E–03	1.30E–01	5.96E–07	1.06E+00	6.57E+00	8.93E–02	2.69E–02
f_{19}	3.82E–04*	3.82E–04*	3.82E–04*	3.82E–04*	3.82E–04*	1.06E+02	3.82E–04*	3.82E–04*
f_{20}	1.13E+00	3.88E–01	2.17E–01	3.21E–01	4.76E–01	4.84E–01	2.14E–01*	3.67E–01
f_{21}	4.99E–01	3.06E–01	3.25E–01	4.38E–01	2.75E–01*	2.97E–01	3.94E–01	4.10E–01
f_{22}	0.00E+00*	7.38E–13	1.62E–03	0.00E+00*	1.92E–04	1.28E+01	3.86E–05	6.16E–02

注：“*”为最优值。

表 3　SLDE与改进算法的优化结果（*D*=100）

Tab. 3　Optimization results of SLDE and improved algorithms with 100 dimensions

函数	平均最优解							
	SLDE	aDE	SHADE	SaDE	CoDE	EPSDE	Rcr–JADE	PLDE
f_1	0.00E+00*	1.25E–14	8.50E–44	8.69E–19	5.33E–11	1.43E+00	5.34E–50	4.68E–36
f_2	0.00E+00*	2.66E–11	1.38E–37	2.52E–15	1.18E–07	7.81E+03	4.18E–44	3.37E–31
f_3	0.00E+00*	1.44E+04	7.14E–02	3.41E+02	7.14E+01	4.19E+05	2.20E–02	1.87E+04
f_4	0.00E+00*	3.80E–09	1.42E–14	1.96E–13	8.10E–06	1.78E+00	6.67E–18	4.60E–19
f_5	7.47E–206*	1.45E+01	6.98E–01	6.43E+00	4.55E–02	4.64E+01	5.37E+00	1.28E+01
f_6	0.00E+00*	4.73E–15	2.27E–43	3.27E–19	2.04E–11	2.10E–01	1.58E–50	4.38E–36
f_7	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*	–1.00E+00*
f_8	0.00E+00*	9.89E–09	2.52E–37	6.69E–13	5.48E–05	7.34E+05	6.10E–44	4.02E–30
f_9	1.10E–302*	1.81E+02	8.14E–05	1.22E+00	9.11E–03	4.12E+03	2.55E–05	8.34E+01
f_{10}	0.00E+00*	9.62E–09	1.31E–35	8.67E–13	4.50E–05	3.18E+05	9.16E–43	8.45E–30
f_{11}	0.00E+00*	0.00E+00*	2.90E+00	0.00E+00*	0.00E+00*	1.33E–01	6.23E+00	8.90E+00
f_{12}	1.36E+04*	1.36E+04*	1.36E+04*	1.36E+04*	1.36E+04*	1.36E+04*	1.36E+04*	1.36E+04*
f_{13}	0.00E+00*	2.35E+02	2.01E–02	6.36E+01	4.66E+02	4.86E+02	9.35E+00	1.97E+02
f_{14}	0.00E+00*	2.47E–04	2.30E–03	3.04E–03	3.60E–11	1.40E–01	3.28E–03	7.75E–03
f_{15}	0.00E+00*	0.00E+00*	1.30E–16	0.00E+00*	1.59E–15	9.35E–05	1.70E–16	6.29E–17
f_{16}	0.00E+00*	3.44E–15	2.01E–44	5.99E–20	1.54E–11	3.32E–01	1.12E–50	6.99E–37
f_{17}	0.00E+00*	1.61E–08	3.11E–01	2.00E+00	9.94E–07	1.39E–01	1.29E+00	1.28E+00
f_{18}	0.00E+00*	6.83E–01	2.56E+01	8.69E+00	3.37E+00	1.48E+02	1.52E+01	1.23E+02
f_{19}	1.27E–03*	1.27E–03*	1.27E–03*	1.27E–03*	1.27E–03*	5.05E+03	1.27E–03*	2.46E+01
f_{20}	1.41E+00	6.06E–01	4.57E–01*	5.52E–01	6.26E–01	7.68E–01	4.97E–01	6.90E–01
f_{21}	5.00E–01*	5.36E–01	5.54E–01	6.07E–01	5.60E–01	1.36E+00	5.92E–01	6.64E–01
f_{22}	0.00E+00*	7.84E–06	2.54E+00	2.64E+00	7.85E–03	9.84E+01	4.54E+00	2.15E+01

注：“*”为最优值。

表 4 Wilcoxon检验结果

Tab. 4 Results of Wilcoxon test

算法	*D*=30				*D*=100			
	R^+	R^-	α=0.05	α=0.1	R^+	R^-	α=0.05	α=0.1
SLDE vs. aDE	111	25	+	+	140	13	+	+
SLDE vs. SHADE	105	31	+	+	174	16	+	+
SLDE vs. SaDE	105	31	+	+	143	10	+	+
SLDE vs. CoDE	124	29	+	+	156	15	+	+
SLDE vs. EPSDE	167	23	+	+	203	7	+	+
SLDE vs. Rcr-JADE	105	31	+	+	177	13	+	+
SLDE vs. PLDE	126	27	+	+	199	11	+	+

注：“+”表示SLDE显著优于其他算法。

表 5 Friedman和Kruskal_Wallis检验结果

Tab. 5 Results of Friedman test and Kruskal_Wallis test

算法	*D*=30		*D*=100	
	Friedman检验平均秩	Kruskal_Wallis检验平均秩	Friedman检验平均秩	Kruskal_Wallis检验平均秩
SLDE	2.48	47.75	1.86	37.82
aDE	5.11	97.02	4.55	89.39
SHADE	3.48	76.52	3.61	83.52
SaDE	4.36	85.57	4.64	90.25
CoDE	5.43	103.25	4.91	89.09
EPSDE	7.11	125.95	7.32	131.86
Rcr-JADE	3.18	76.02	3.75	85.66
PLDE	4.84	95.91	5.36	100.41

表 6 平均迭代次数和实验成功率(*D*=30)

Tab. 6 Mean iterations and success rate with 30 dimensions

函数	SLDE		aDE		SHADE		SaDE		CoDE		EPSDE		Rcr-JADE		PLDE	
	MEGN	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%
f_1	24	100	746	100	266	100	447	100	1 064	100	1 366	80	250	100	321	100
f_2	28	100	931	100	356	100	568	100	1 313	100	1 407	3.3	332	100	416	100
f_3	32	100	—	0	703	100	—	0	—	0	—	0	690	100	—	0
f_4	29	100	1 025	100	437	100	654	100	—	0	—	0	407	100	474	100
f_5	53	100	—	0	741	100	—	0	—	0	—	0	765	100	—	0
f_6	22	100	694	100	252	100	415	100	981	100	1 336	76.7	234	100	296	100
f_7	18	100	493	100	184	100	296	100	699	100	955	100	173	100	213	100
f_8	33	100	1 102	100	377	100	657	100	—	0	—	0	353	100	471	100
f_9	64	100	—	0	871	100	—	0	—	0	—	0	830	100	—	0
f_{10}	32	100	1 090	100	400	100	659	100	—	0	—	0	364	100	473	100
f_{11}	10	100	267	100	115	100	161	100	389	100	554	100	105	100	125	100
f_{12}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{13}	18	100	—	0	1 220	100	1 080	100	—	0	—	0	1 057	100	—	0
f_{14}	19	100	490	100	184	100	300	96.7	787	100	1 065	93.3	172	86.7	246	83.3
f_{15}	8	100	199	100	85	100	115	100	266	100	394	100	77	100	85	100
f_{16}	15	100	415	100	163	100	252	100	593	100	821	100	149	100	179	100
f_{17}	25	100	560	100	208	100	329	100	801	100	1 086	96.7	195	100	243	100
f_{18}	59	100	—	0	—	0	1 046	100	—	0	—	0	797	3.3	867	13.3
f_{19}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{20}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{21}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{22}	23	100	465	100	243	100	312	100	756	100	—	0	229	100	231	90

注：“—”表示未测得。

表 7　平均迭代次数和实验成功率(*D*=100)

Tab. 7　Mean iterations and success rate with 100 dimensions

函数	SLDE		aDE		SHADE		SaDE		CoDE		EPSDE		Rcr-JADE		PLDE	
	MEGN	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%	*MEGN*	*SR*/%
f_1	27	100	2 063	100	583	100	1 595	100	2 545	100	—	0	620	100	961	100
f_2	32	100	2 574	100	949	100	2 069	100	—	0	—	0	921	100	1 314	100
f_3	39	100	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_4	32	100	2 875	100	1 508	100	1 992	100	—	0	—	0	1 333	100	1 499	100
f_5	60	100	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_6	26	100	1 996	100	577	100	1 533	100	2 463	100	—	0	613	100	943	100
f_7	22	100	1 385	100	387	100	1 019	100	1 695	100	2 662	6.7	415	100	644	100
f_8	36	100	2 953	73.3	941	100	2 405	100	—	0	—	0	963	100	1 403	100
f_9	92	100	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{10}	36	100	2 954	73.3	1 040	100	2 408	100	—	0	—	0	1 006	100	1 430	100
f_{11}	13	100	769	100	799	100	1 001	100	916	100	2 316	96.7	—	0	438	3.3
f_{12}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{13}	22	100	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{14}	19	100	1 255	96.7	355	80	900	76.7	1 528	100	2 666	6.7	374	70	580	83.3
f_{15}	11	100	609	100	169	100	385	100	718	100	1 895	96.7	174	100	278	100
f_{16}	18	100	1 202	100	332	100	857	100	1 460	100	2 838	6.7	354	100	551	100
f_{17}	29	100	1 487	100	459	100	—	0	1 817	100	2 909	3.3	519	6.7	740	10
f_{18}	72	100	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{19}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{20}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{21}	—	0	—	0	—	0	—	0	—	0	—	0	—	0	—	0
f_{22}	31	100	1 534	100	—	0	—	0	2 232	100	—	0	—	0	—	0

注：“—”表示未测得。

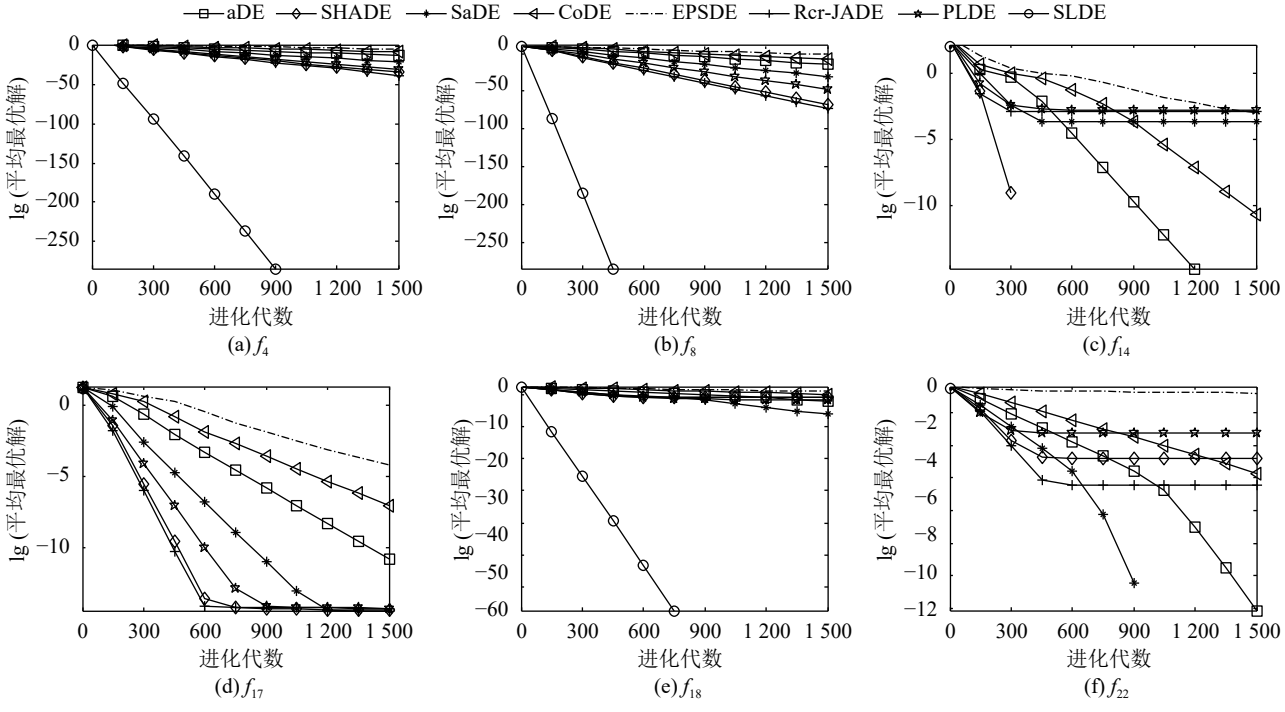

图 1　平均最优解的进化曲线(*D*=30)

Fig. 1　Evolution curves of the mean best solution with 30 dimensions

4 结 论

针对传统微分进化算法早熟收敛导致收敛精度低问题，作者提出基于共享学习变异策略的微分进化算法（SLDE）。算法引入共享个体，通过最优个体与共享个体的差分实现整个种群之间的信息共享，提高了算法的搜索能力；根据个体的适应值自适应地调整共享学习因子，充分利用个体的进化信息，改善了种群进化的趋利性。实验结果表明：无论是单峰问题还是多峰问题，SLDE在收敛精度和收敛速度方面均优于其他改进的DE算法，具备更好的实时性和稳定性。

提出的SLDE算法采用固定的控制参数，未对控制参数策略进行研究，未来研究工作将关注控制参数的自适应调整策略，进一步提高算法的优化性能，以解决实际应用中更为复杂的优化问题。

参考文献：

[1] Storn R,Price K.Differential evolution:A simple and efficient adaptive scheme for global optimization over continuous spaces[R].Berkeley:University of California,Berkeley,1995.

[2] Arce F, Zamorab E, Sossaa H, et al.Differential evolution training algorithm for dendrite morphological neural networks[J].Applied Soft Computing,2018,68:303–313.

[3] Ghosh A, Datta A, Ghosh S.Self-adaptive differential evolution for feature selection in hyperspectral image data[J].Applied Soft Computing,2013,13(4):1969–1977.

[4] Kok K Y, Rajendran P.Differential-evolution control parameter optimization for unmanned aerial vehicle path planning[J].PLOS ONE,2016,11(3):e0150558.

[5] Wang Haiyan, Zhang Yanwei, Zhao Jingling, et al.Batch optimized scheduling of intermingling flow-shop based on hybrid differential evolution algorithm[J].Computer Integrated Manufacturing Systems,2013,19(7):1613–1625.

[6] Marčič T, Štumberger B, Štumberger G.Differential-evolution-based parameter identification of a line-start IPM synchronous motor[J].IEEE Transactions on Industrial Electronics,2014,61(11):5921–5929.

[7] Kadhar K M A, Baskar S, Amali S M J.Diversity controlled self-adaptive differential evolution based design of non-fragile multivariable PI controller[J].Engineering Applications of Artificial Intelligence,2015,46(PA):209–222.

[8] Liu J, Lampinen J.A fuzzy adaptive differential evolution algorithm[J].Soft Computing,2005,9(6):448–462.

[9] Brest J, Greiner S, Boskovic B, et al.Self-adapting control parameters in differential evolution:A comparative study on numerical Benchmark problems[J].IEEE Transactions on Evolutionary Computation,2006,10(6):646–657.

[10] Noman N,Bollegala D,Iba H.An adaptive differential evolution algorithm[C]//Proceedings of the 2011 IEEE Congress on Evolutionary Computation.New Orleans:IEEE,2011:2229–2236.

[11] Tanabe R,Fukunaga A.Success-history based parameter adaptation for differential evolution[C]//Proceedings of the 2013 IEEE Congress on Evolutionary Computation.Cancun:IEEE,2013,71–78.

[12] Zhang Jingqiao, Sanderson A C.JADE:Adaptive differential evolution with optional external archive[J].IEEE Transactions on Evolutionary Computation,2009,13(5):945–958.

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

[14] Wang Yong, Cai Zixing, Zhang Qingfu.Differential evolution with composite trial vector generation strategies and control parameters[J].IEEE Transactions on Evolutionary Computation,2011,15(1):55–66.

[15] 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.

[16] Wu Guohua, Shen Xin, Li Haifeng, et al.Ensemble of differential evolution variants[J].Information Sciences,2018,423:172–186.

[17] Gong Wenyin, Cai Zhihua, Wang Yang.Repairing the crossover rate in adaptive differential evolution[J].Applied Soft Computing,2014,15:149–168.

[18] Liu Hao, Ding Jinliang, Yang Cui'e, et al.Perferred-learning-based differential evolution algorithm[J].Journal of Shanghai Jiaotong University,2017,51(6):704–708.[刘昊, 丁进良, 杨翠娥, 等.基于择优学习策略的差分进化算法[J].上海交通大学学报,2017,51(6):704–708.]

[19] Wang Shihao, Li Yuzhen, Yang Hongyu.Self-adaptive differential evolution algorithm with improved mutation mode[J].Applied Intelligence,2017,47(3):644–658.

（编辑 赵 婧）

引用格式：Duan Meijun,Yang Hongyu,Liu Hong,et al.Differential evolution algorithm based on sharing learning[J].Advanced Engineering Sciences,2019,51(1):205–212.[段美军,杨红雨,刘洪,等.基于共享学习策略的微分进化算法[J].工程科学与技术,2019,51(1):205–212.]