计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2014.20130395
51(10):2187-2194,2014

基于阈值统计学习的差分进化引力搜索算法

张英杰　　龚中汉
(湖南大学信息科学与工程学院　长沙　410082)
(zhangyj@hnu.edu.cn)

Hybrid Differential Evolution Gravitation Search Algorithm Based on Threshold Statistical Learning

Zhang Yingjie and Gong Zhonghan
(*College of Information Science and Engineering*, *Hunan University*, *Changsha* 410082)

Abstract　Differential evolution (DE) is a simple and efficient population-based stochastic real-parameter optimization algorithm, which has been applied to a wide range of complex optimization problems. However, the standard DE has some drawbacks, such as premature convergence, low convergence precision and slow convergence rate in the later stage of evolution. To deal with these drawbacks, a novel hybrid differential evolution algorithm (DEGSA-SL) is proposed by combining the advantage of gravitational search algorithm (GSA). In the proposed algorithm, a new operator called threshold statistical learning is designed. By introducing the operator, the better strategy of DE and GSA can be selected adaptively, by learning from the previous success ratio of the two strategies, to produce next generation at each iteration in the evolution process. It takes full use of the potential of DE and GSA, ensures the balance between global exploration and local exploitation abilities in the solution spaces, and improves the global search capabilities of the standard DE algorithm. Several complex benchmark functions are employed to test the performance of the DEGSA-SL. The results show that the proposed algorithm not only achieves better convergence precision, robustness and convergence rate, but also avoids the premature convergence problem effectively.

Key words　differential evolution (DE); gravitational search algorithm (GSA); threshold statistical learning; hybrid algorithm; numerical optimization

摘　要　为了改善基本差分进化算法在求解复杂优化问题时易出现早熟收敛、求解精度低以及进化后期收敛速度慢等缺陷,结合引力搜索算法的优点,提出一种基于阈值统计学习思想的混合差分进化引力搜索算法.该算法通过阈值统计学习的方式,充分利用差分进化算法的全局优化能力与引力搜索算法在进化后期的种群开发能力,在进化过程中根据2种策略在先前学习代数的成功率自适应选择较优策略生成下一代群体,保证种群在解空间中的探索与开发能力之间的平衡,以提高算法的全局寻优能力.对几个经典复杂测试函数的仿真结果表明:改进算法求解精度高、收敛速度快、鲁棒性强、能够有效避免早熟收敛问题.

关键词　差分进化;引力搜索算法;阈值统计学习;混合算法;数值优化

中图法分类号　TP18

收稿日期:2013-03-29;**修回日期**:2013-12-11

基金项目:国家自然科学基金项目(61174140);湖南省自然科学基金重点项目(13JJA002)

差分进化算法(differential evolution, DE)是由 Storn 和 Price 于 1995 年提出的一种基于种群的全局优化方法[1-2]. 该算法实现简单、控制参数少且优化性能较好,现已成功应用于科学与工程领域[3-4]. 然而,基本 DE 算法的种群开发能力较弱,以致于算法在迭代后期收敛速度减慢,易陷入局部最优. 近年来,国内外相关学者提出了多种改进措施,如算法控制参数的改进[5-6]、变异策略的改进[7-8]以及混合差分进化算法[9-11]等. 其中,混合差分进化算法是目前的一个研究热点. Chen 等人基于统计学习提出一种混合差分进化粒子群(DEPSO)算法,该算法在迭代过程中根据两种策略在先前学习代数中的相对成功率来选择算法下一步的进化策略[9];Das 等人对基本 DE 算法中选择策略进行改进,融合模拟退火的思想,以一定的概率接受较差解[10];Li 等人在基本 DE 算法的选择过程中,融入引力搜索策略来改善实验个体的质量[11]. 上述改进虽然在一定程度上改善了算法性能,但是早熟收敛问题依然存在,而且一些改进措施还增加了算法的函数评价次数及时间复杂度.

引力搜索算法(gravitational search algorithm, GSA)是 Rashedi 教授于 2009 年基于万有引力定律而提出的一种新型优化算法[12]. 该算法主要通过个体间的万有引力来实现信息的共享,由较优个体集对当前个体的合力来指导个体的搜索方向. 由于算法在迭代过程中,较优个体的数目随着迭代次数的增加而减少,并最终趋于全局最优个体,因此 GSA 在迭代后期收敛速度快,具有较强的种群开发能力.

针对基本 DE 算法的缺陷,本文充分考虑 GSA 在迭代后期的优势,设计了一种阈值统计学习算子,提出了一种基于阈值统计学习思想的混合差分进化引力搜索算法(DEGSA-SL). 该混合算法旨在利用 DE 与 GSA 两种算法的优点,扬长避短,并通过阈值统计学习的方式保证整个种群在探索与开发能力之间的平衡,以期望改善基本 DE 算法的收敛性能.

1 差分进化算法与引力搜索算法

1.1 差分进化算法

DE 算法[1-2]是一种基于种群的随机搜索算法,具有实现简单、控制参数少、鲁棒性强等特点. 标准的 DE 算法采用实数编码,由 NP 个 D 维个体 $\boldsymbol{X}_i(t)=\{x_i^1,\cdots,x_i^D\}$, $i=1,\cdots,NP$ 构成种群,在进化过程中通过变异、交叉、选择等操作生成下一代群体来搜索整个解空间,最终使种群在迭代完成时获得满意解或最优解.

1) 变异操作. 对种群中每个个体 $\boldsymbol{X}_i(t)$,根据式(1)产生变异个体 $\boldsymbol{V}_i(t)=\{v_i^1,\cdots,v_i^D\}$:

$$\boldsymbol{V}_i(t)=\boldsymbol{X}_{r1}(t)+F\times(\boldsymbol{X}_{r2}(t)-\boldsymbol{X}_{r3}(t)), \quad (1)$$

其中,下标 $i,r1,r2,r3\in\{1,2,\cdots,NP\}$ 为互不相等的整数;F 为比例因子,用于控制差分向量的幅度.

2) 交叉操作. 为增加种群多样性,将种群中每个个体 $\boldsymbol{X}_i(t)$ 与其对应的变异个体 $\boldsymbol{V}_i(t)$,按一定的交叉概率 CR 进行交叉,生成实验个体 $\boldsymbol{U}_i(t)=\{u_i^1,\cdots,u_i^D\}$:

$$u_i^j(t)=\begin{cases}v_i^j(t), & \text{if } rand_j[0,1)\leqslant CR,\ j=j_{\text{rand}},\\ x_i^j(t), & \text{otherwise},\end{cases} \quad (2)$$

其中,交叉概率 $CR\in[0,1]$,j_{rand} 为区间 $[1,D]$ 中随机选取的整数. 这种交叉方式可以保证实验个体 $\boldsymbol{U}_i(t)$ 中至少有一个分量来自变异个体.

3) 选择操作. DE 算法通过一对一的贪婪选择策略来生成下一代群体,即

$$\boldsymbol{X}_i(t+1)=\begin{cases}\boldsymbol{U}_i(t), & \text{if } f(\boldsymbol{U}_i(t))\leqslant f(\boldsymbol{X}_i(t)),\\ \boldsymbol{X}_i(t), & \text{otherwise},\end{cases} \quad (3)$$

其中,$f(*)$ 为目标函数.

1.2 引力搜索算法

与 DE 算法类似,GSA 也是一种基于种群的随机搜索算法[12],但是,GSA 的理论基础是牛顿万有引力定律——"自然界中的任何粒子都是相互吸引的,且引力大小与 2 个粒子间质量的乘积成正比,与 2 个粒子间距离的平方成反比". 在进化过程中,GSA 利用种群中每个个体 $X_i(t)$ 的适应度值计算出其对应的质量 $M_i(t)$,个体的质量越大,就意味着它越接近全局最优,且对其他个体的吸引力越大. 因此,较优个体将吸引其他个体向它靠拢,最终所有个体都趋向于全局最优.

针对由 NP 个 D 维个体 $\boldsymbol{X}_i(t)=\{x_i^1,\cdots,x_i^D\}$, $i=1,\cdots,NP$ 构成的种群,GSA 从某一随机生成的初始种群开始,根据式(4)计算出所有个体在每一维上的相互吸引力:

$$\boldsymbol{F}_{ij}^d(t)=G(t)\frac{\boldsymbol{M}_i(t)\times\boldsymbol{M}_j(t)}{\boldsymbol{R}_{ij}(t)+\varepsilon}(x_j^d(t)-x_i^d(t)), \quad (4)$$

式(4)中,$\boldsymbol{M}_i(t)$ 与 $\boldsymbol{M}_j(t)$ 分别为个体 i 和个体 j 的质量;ε 是一个很小的常量;$G(t)$ 表示万有引力常量;$\boldsymbol{R}_{ij}(t)$ 表示个体 i 和个体 j 之间的欧氏距离,且:

$$G(t)=G_0\times\exp(-\alpha\times t/iter_{\max}), \quad (5)$$

$$\boldsymbol{R}_{ij}(t)=\|\boldsymbol{X}_i(t),\boldsymbol{X}_j(t)\|_2, \quad (6)$$

其中,G_0 为初始万有引力常量;α 为下降系数;

$iter_{\max}$为算法的最大迭代次数.

为增加算法的随机性,个体 i 在第 d 维所受合力定义为

$$\boldsymbol{F}_i^d(t)=\sum_{j=1,j\neq i}^{NP} rand_j \boldsymbol{F}_{ij}^d(t), \tag{7}$$

其中,$rand_j\in[0,1]$;$\boldsymbol{F}_{ij}^d(t)$为个体 j 与个体 i 之间的万有引力.

根据牛顿第 2 定律,个体 i 在第 t 代的加速度定义为

$$a_i^d(t)=\frac{\boldsymbol{F}_i^d(t)}{\boldsymbol{M}_i(t)}, \tag{8}$$

其中,$\boldsymbol{M}_i(t)$为个体 i 在第 t 代的质量.

最后,GSA 在进化过程中根据式(9)、式(10)更新个体的速度与位置:

$$v_i^d(t+1)=rand_i\times v_i^d(t)+a_i^d(t), \tag{9}$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1), \tag{10}$$

其中,$rand_i\in[0,1]$.

在 GSA 中,个体 i 在第 t 代的质量定义为

$$m_i(t)=\frac{fit_i(t)-worst(t)}{best(t)-worst(t)}, \tag{11}$$

$$\boldsymbol{M}_i(t)=\frac{m_i(t)}{\sum_{j=1}^{NP} m_j(t)}, \tag{12}$$

其中,$fit_i(t)$为个体 i 在第 t 代的适应度值;$best(t)$与 $worst(t)$分别为第 t 代时种群中的最优个体与最差个体的适应度值.

对于求最小值问题,$best(t)$与 $worst(t)$的定义为

$$best(t)=\min_{j\in\{1,\cdots,NP\}} fit_j(t), \tag{13}$$

$$worst(t)=\max_{j\in\{1,\cdots,NP\}} fit_j(t), \tag{14}$$

反之,即可用于求最大值问题.

2 DEGSA-SL 算法设计及实现

2.1 阈值统计学习

统计学习[9,13]是指算法在迭代过程中根据每种策略的成功率依概率自适应选择个体下一代的进化策略.例如对于本文算法,假定种群在第 t 代中个体选择 DE 策略的概率为 $p_{DE}(t)$,选择 GSA 策略的概率为 $p_{GSA}(t)=1-p_{DE}(t)$,种群在第 t 代中的最优解通过 DE 和 GSA 成功改善的次数分别记为 $ns_{DE}(t)$与 $ns_{GSA}(t)$,失败次数分别记为 $nf_{DE}(t)$与 $nf_{GSA}(t)$.因此,在第 t 代中个体选择 DE 策略的概率定义为

$$p_{DE}(t)=\frac{S_{DE}(t)}{S_{DE}(t)+S_{GSA}(t)}, \tag{15}$$

$$S_k(t)=\frac{\sum_{iter=t-LP}^{t-1} ns_k(t)}{\sum_{iter=t-LP}^{t-1} ns_k(t)+\sum_{iter=t-LP}^{t-1} nf_k(t)+\varepsilon}+\varepsilon, \tag{16}$$

其中,$S_k(t)$为第 k 种策略的成功率;$k\in\{DE,GSA\}$;LP 为学习代数,且 $t>LP$;ε 为一个很小的常数,以保证每种策略的成功率及选择概率不为 0.

统计学习方法虽然能够利用各种策略的优点来改善算法性能,但是这种利用是有限的.在进化过程中,当种群采取某种策略后而陷入局部最优或在局部最优点周围振荡时,该策略尽管能改善当前最优解,但这种改善会使该策略的选择概率趋向于 1,而且如果这种改善程度很小,不足以帮助算法跳出局部最优,则算法就会收敛于该局部最优点,产生早熟收敛.

因此,本文提出一种阈值统计学习的方法,以尽量避免上述问题的发生.其主要思想是,只有当第 t 代种群中个体通过某一策略而新产生的解 $curVal$ 与种群最优解 $bestVal$ 之间的改善程度 imp 大于预定义的阈值 δ 时,算法才会增加该策略的成功次数,否则增加该策略的失败次数.其中,改善程度 imp 定义为

$$imp=(bestVal-curVal)/bestVal. \tag{17}$$

2.2 DEGSA-SL 算法思想

类似于反馈控制系统,DEGSA-SL 算法通过阈值统计学习的方式,对 2 种策略在前 LP 代中的成功率进行计算,并将这一信息反馈给当前种群,使整个种群逐渐趋于选择成功率较高的策略作为下一代的进化策略,最终收敛于全局最优点.而且由于算法在进化过程中的不断学习,使种群在每一次的迭代中都以较大概率选择相对较优的策略,充分发掘了 DE 与 GSA 的潜力,保证了种群在探索与开发能力之间的平衡,提高了算法的收敛速度与收敛精度.

在 DEGSA-SL 中,除了 DE 与 GSA 本身的参数以外,主要引入了 2 个参数:学习代数 LP 与阈值 δ.对于学习代数 LP,较小的学习代数会增加算法的随机性,使统计结果不可信,而较大的学习代数则会降低算法的收敛速度与收敛精度.同样,对于阈值 δ,较小的阈值会增加算法陷入局部最优的风险,而较大的阈值则会使统计结果变得不可信.因此,本文针对上述 2 个参数进行了分析,以保证算法的有效性.

2.3 DEGSA-SL 算法流程

根据 2.2 节算法思想,本文提出的 DEGSA-SL

算法流程描述如算法 1 所示：

算法 1. DEGSA-SL 算法.

Step1. 随机初始化 NP 个个体的位置，并初始化相关参数；

Step2. 计算每个个体的适应值，保存初始最优位置与初始最优适应值；

Step3.

While (不满足结束条件)Do /* 结束条件为找到可接受值或达到设定的最大函数评价次数 */

If 迭代代数 $t>LP$

根据式(15)(16)更新种群在第 t 代选择 DE 策略的概率 $p_{DE}(t)$；

End If

For 种群中的每一个个体

If $rand<p_{DE}(t)$ /* 算法根据 DE 策略产生新个体 */

根据式(1)～(3)计算新产生个体的位置，并对其进行评价，如果新个体适应度值大于当前最优值，则更新当前最优值；根据式(17)计算新产生解的改善程度 imp，如果改善程度 imp 大于阈值 δ，则 $ns_{DE}(t)=ns_{DE}(t)+1$；否则 $nf_{DE}(t)=nf_{DE}(t)+1$；

Else /* 算法根据 GSA 策略产生新个体 */

根据式(9)(10)及相应公式计算新产生个体的位置与速度，并对其进行评价，如果新个体适应值大于当前最优值，则更新当前最优值；根据式(17)计算新产生解的改善程度 imp，如果改善程度 imp 大于阈值 δ，则 $ns_{GSA}(t)=ns_{GSA}(t)+1$；否则 $nf_{GSA}(t)=nf_{GSA}(t)+1$；

End If

End For

End While

Step4. 输出结果，算法结束.

3 仿真实验与结果分析

实验仿真环境为 Windows XP 系统；Intel Pentium® CPU 2.70 GHz；内存 4.0 GB；仿真软件为 Visual C++ 6.0. 为验证本文提出的混合算法 DEGSA-SL 对复杂函数优化时的收敛速度与收敛精度等性能，引入若干基准测试函数对算法进行测试，以全面考察算法对于不同类型优化问题的性能. 文中基准测试函数均来自文献[14]，其中 $f_1\sim f_4$ 为单模态函数，$f_5\sim f_8$ 为多模态函数. 具体如下：

$$f_1=\sum_{i=1}^{D}z_i^2,\ \boldsymbol{Z}=\boldsymbol{X}-\boldsymbol{O};$$

$$f_2=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}z_j\right)^2,\ \boldsymbol{Z}=\boldsymbol{X}-\boldsymbol{O};$$

$$f_3=\sum_{i=1}^{D-1}(100(x_i^2-x_{i+1})^2+(x_i-1)^2);$$

$$f_4=\left(\sum_{i=1}^{D}\left(\sum_{j=1}^{i}z_j\right)^2\right)(1+0.4\mid N(0,1)\mid),\ \boldsymbol{Z}=\boldsymbol{X}-\boldsymbol{O};$$

$$f_5=\sum_{i=1}^{D}\frac{z_i^2}{4000},\ \boldsymbol{Z}=\boldsymbol{X}-\boldsymbol{O};$$

$$f_6=\sum_{i=1}^{D}\frac{z_i^2}{4000},\ \boldsymbol{Z}=\boldsymbol{M}(\boldsymbol{X}-\boldsymbol{O}),cond(\boldsymbol{M})=3;$$

$$f_7=\sum_{i=1}^{D}(z_i^2-10\cos(2\pi z_i)+10),\ \boldsymbol{Z}=\boldsymbol{X}-\boldsymbol{O};$$

$$f_8=\sum_{i=1}^{D}(z_i^2-10\cos(2\pi z_i)+10),\ \boldsymbol{Z}=\boldsymbol{M}(\boldsymbol{X}-\boldsymbol{O}),cond(\boldsymbol{M})=2.$$

其中，$\boldsymbol{O}=[o_1,o_2,\cdots,o_D]$为移位后的全局最优点，上述函数的详细描述如表 1 所示：

Table 1 Test Functions

表 1 测试函数

Test Functions	Initialization Range	Dimension	Global Optimum $\boldsymbol{X}$	Acceptance	Global Optimum
f_1	$-100\leqslant X_i\leqslant 100$	30	$\boldsymbol{O}$	0.000 01	0
f_2	$-100\leqslant X_i\leqslant 100$	30	$\boldsymbol{O}$	0.000 01	0
f_3	$-100\leqslant X_i\leqslant 100$	30	(1,1,…,1)	20	0
f_4	$-100\leqslant X_i\leqslant 100$	30	$\boldsymbol{O}$	5 000	0
f_5	$0\leqslant X_i\leqslant 600$	30	$\boldsymbol{O}$	0.000 01	0
f_6	$0\leqslant X_i\leqslant 600$	30	$\boldsymbol{O}$	0.1	0
f_7	$-5\leqslant X_i\leqslant 5$	30	$\boldsymbol{O}$	60	0
f_8	$-5\leqslant X_i\leqslant 5$	30	$\boldsymbol{O}$	50	0

实验参数设置:种群个体数设为 50,最大函数评价次数(FEs_{max})设为 300 000[14],为减少实验误差,所有实验独立运行 50 次,取平均结果进行比较.文中对比算法 DE/rand/1 算法参数设置为 $F=0.5$,$CR=0.3$,GSA 算法[12]与 DE-GSA 算法[11]分别参照各自文献.在本文 DEGSA-SL 算法中,$F=0.5$,$CR=0.3$,初始概率 p_{DE} 设为 0.5,学习代数 LP 与阈值 δ 分别设为 20 和 0.15.

3.1 对比实验

表 2 中 *Mean* 与 *Std. Dev* 分别表示适应度的均值与标准差.表 3 中 *SR* 表示 50 次独立运行中算法收敛精度均达到可接受值的成功率,*NFEs* 表示 *SR* 为 100%时算法的平均函数评价次数.从表 2、表 3 及图 1 可以看出,在函数 f_1 上,DEGSA-SL 算法的收敛精度与 DE/rand/1 算法,DE-GSA 算法一样,均能达到全局最优,但 DEGSA-SL 算法的收敛速度却快于这 2 种算法;在函数 f_2,f_3,f_6 上,DEGSA-SL 算法的收敛精度与收敛速度均优于其他算法;在函数 f_4 上,DEGSA-SL 算法的收敛精度及成功率均优于两种基本算法,而 DE-GSA 算法的性能则优于其他 3 种算法;在函数 f_5 上,DEGSA-SL 算法的收敛精度与 DE/rand/1 算法一样,均能达到全局最优,但 DEGSA-SL 算法的收敛速度却比 DE/rand/1 算法快;在函数 f_7 上,根据图 1(g)可知,DE 算法与 GSA 算法在进化过程中的优化能力较为接近,导致 DEGSA-SL 算法中 2 种策略的选择出现随机现象,以致于算法的收敛精度比基本算法差,而对于 DE-GSA 算法能达到全局最优,主要是由于其中 DE 算法的参数 F 与 CR 的选择问题;在函数 f_8 上,DEGSA-SL 算法的收敛精度及成功率均优于其他 3 种算法.上述结果表明,无论是单模态还是多模态问题,只要在该问题中 2 种基本算法的寻优能力相差较大,则 DEGSA-SL 算法的寻优能力通常就会强于 2 种基本算法.

Table 2 Comparison of DEGSA-SL and the Others (Convergence Precision)

表 2 DEGSA-SL 算法与其他 3 种算法收敛精度比较

Test Function	DE/rand/1		GSA		DE-GSA		DEGSA-SL	
	Mean	*Std. Dev*	*Mean*	*Std. Dev*	*Mean*	*Std. Dev*	*Mean*	*Std. Dev*
f_1	**0**	**0**	3.04E−17	7.58E−18	**0**	**0**	**0**	**0**
f_2	5.87E+03	1.67E+03	2.15E−16	8.83E−17	1.31E−06	3.24E−07	**1.67E−16**	**7.40E−17**
f_3	3.67E+01	2.30E+01	3.04E+01	2.41E+01	2.44E+01	2.23E+01	**1.41E+01**	**1.49E+00**
f_4	1.49E+04	3.39E+03	3.55E+03	1.82E+03	**8.17E+01**	**2.72E+02**	1.81E+03	2.44E+03
f_5	**0**	**0**	1.20E+03	7.56E+01	9.14E+01	1.62E+02	**0**	**0**
f_6	1.52E−02	2.24E−02	4.04E+03	2.62E+02	5.09E+02	4.81E+02	**9.80E−03**	**8.26E−03**
f_7	3.37E+01	3.21E+00	2.67E+01	2.33E+01	**0**	**0**	4.60E+01	5.43E+00
f_8	1.87E+02	9.70E+00	3.61E+01	3.78E+01	4.92E+01	1.65E+01	**2.07E+01**	**2.89E+01**

Note: Boldface represent the best result in the test functions.

Table 3 Comparison of DEGSA-SL and the Others (Convergence Rate)

表 3 DEGSA-SL 算法与其他 3 种算法收敛速度比较

Test Function	DE/rand/1		GSA		DE-GSA		DEGSA-SL	
	NFEs	*SR*/%	*NFEs*	*SR*/%	*NFEs*	*SR*/%	*NFEs*	*SR*/%
f_1	31 511	100	132 202	100	45 403	100	**27 972**	**100**
f_2	—	0	142 076	100	269 687	100	**140 709**	**100**
f_3	—	8	—	8	—	70	**109 296**	**100**
f_4	—	0	—	78	**24 696**	**100**	—	92
f_5	35 279	100	—	0	—	52	**32 950**	**100**
f_6	139 447	100	—	0	—	0	**54 766**	**100**
f_7	112 067	100	—	98	**34 434**	**100**	190 166	100
f_8	—	0	—	84	—	42	—	**88**

Note: Boldface represent the best result in the test functions.

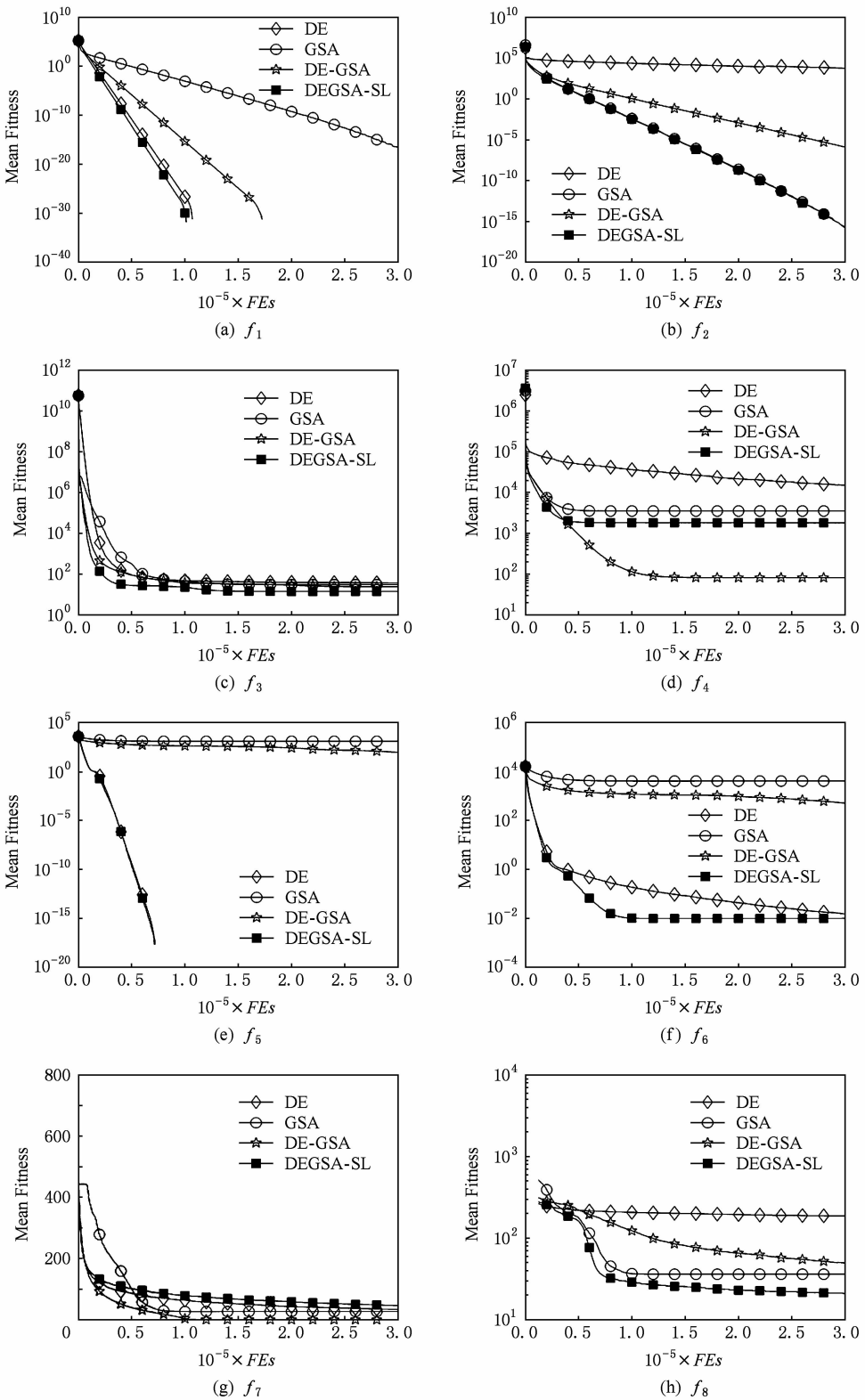

Fig. 1 Median convergence characteristics of DEGSA-SL and the others on 8 test functions.

图 1 DEGSA-SL 算法与其他 3 种算法的收敛性能比较

3.2 算法参数分析

在 DEGSA-SL 算法中,为有效地获取学习代数 LP 与阈值 δ,本文分别使用 3 个单模态函数 f_2,f_3 与 f_4 以及 3 个多模态函数 f_5,f_6 与 f_8 来分析不同参数值对算法性能的影响. 表 4 与表 5 中 Rank 表示在该问题中不同学习代数 LP 或阈值 δ 所对应的算法收敛精度排名. 总排名(total rank)越小,表示算法在该参数值下的性能越好. 从表 4 与表 5 可

Table 4 Effects of LP on the Performance of DEGSA-SL ($\delta=0.015$)

表 4 学习代数 LP 对 DEGSA-SL 算法性能的影响(阈值 $\delta=0.015$)

Function	Evaluation Index	LP				
		10	20	30	40	50
f_2	Mean	**1.35E−16**	1.67E−16	1.89E−16	3.40E−16	3.43E−16
	Rank	**1**	2	3	4	5
f_3	Mean	1.55E+01	**1.41E+01**	1.46E+01	1.53E+01	1.50E+01
	Rank	5	**1**	2	4	3
f_4	Mean	3.27E+03	**1.81E+03**	2.35E+03	2.40E+03	2.61E+03
	Rank	5	**1**	2	3	4
f_5	Mean	8.54E−03	**0**	**0**	**0**	**0**
	Rank	5	**1**	**1**	**1**	**1**
f_6	Mean	5.83E−01	**9.80E−03**	1.12E−02	1.19E−02	1.17E−02
	Rank	5	**1**	2	4	3
f_8	Mean	2.09E+01	**2.07E+01**	2.20E+01	2.37E+01	2.74E+01
	Rank	2	**1**	3	4	5
$f_2\sim f_6, f_8$	Total Rank	23	7	13	20	21

Note: Boldface represent the best result in the test functions.

Table 5 Effects of δ on the Performance of DEGSA-SL ($LP=20$)

表 5 阈值 δ 对 DEGSA-SL 算法性能的影响(学习代数 $LP=20$)

Function	EvaluationIndex	δ					
		0	0.005	0.01	0.015	0.02	0.025
f_2	Mean	1.80E−16	1.87E−16	1.92E−16	**1.67E−16**	1.86E−16	1.70E−16
	Rank	3	5	6	**1**	4	2
f_3	Mean	1.96E+01	1.51E+01	1.51E+01	**1.41E+01**	1.81E+01	1.50E+01
	Rank	6	3	3	**1**	5	2
f_4	Mean	1.91E+03	2.26E+03	2.18E+03	**1.81E+03**	3.63E+03	3.73E+03
	Rank	2	4	3	**1**	5	6
f_5	Mean	**0**	**0**	**0**	**0**	3.45E−04	1.48E−04
	Rank	**1**	**1**	**1**	**1**	6	5
f_6	Mean	1.31E−01	2.66E−01	1.20E−02	**9.80E−03**	1.13E−02	1.37E−02
	Rank	4	6	3	**1**	2	5
f_8	Mean	2.16E+01	2.46E+01	2.76E+01	**2.07E+01**	2.39E+01	2.66E+01
	Rank	2	4	6	**1**	3	5
$f_2\sim f_6, f_8$	Total Rank	18	23	22	**6**	25	25

Note: Boldface represent the best result in the test functions.

以看出,学习代数 LP 有 5 个取值,阈值 δ 有 6 个取值.因此,总共有 30 组不同的参数组合.为减少篇幅,文中选取表 4 与表 5 来对参数进行分析.

从表 4 可以看出,阈值 δ 固定为 0.015,当 $LP=20$ 时,除函数 f_2 以外,算法收敛精度都达到最优,且总排名最小,因此选择 LP 为 20.从表 5 可以看出,学习代数 LP 固定为 20,当阈值 $\delta=0.015$ 时,对所有函数,算法收敛精度都达到最优,且总排名最小,因此选择阈值 $\delta=0.015$.

4 结 论

基本 DE 算法全局搜索能力较强,但存在收敛速度慢、易早熟收敛等缺陷,而 GSA 在迭代后期有

较强的种群开发能力与快速收敛能力.因此,在充分利用 DE 与 GSA 优点的基础上,采用阈值统计学习的方式,使种群在进化过程中能够自适应选择较优策略生成子种群,保证了种群在探索与开发能力之间的平衡,有效地提高了改进混合搜索算法的性能.仿真实验结果表明,本文所提出的差分进化引力搜索算法具有较好的收敛速度与收敛精度,能有效避免早熟收敛,而且鲁棒性强,是一种有效的全局优化算法.

参 考 文 献

[1] Storn R, Price K. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, TR-95-012 [R]. Berkeley: International Computer Science Institute (ICSI), 1995

[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] Wang Ling, Xu Ye, Li Lingpo. Parameter identification of chaotic systems by hybrid Nelder-Mead simplex search and differential evolution algorithm [J]. Expert Systems with Applications, 2011, 38(4): 3238-3245

[4] Noman N, Iba H. Differential evolution for economic load dispatch problems [J]. Electric Power Systems Research, 2008, 78(8): 1322-1331

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

[6] Islam S, Das S, Ghosh S, et al. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization [J]. IEEE Trans on Systems, Man and Cybernetics, Part B: Cybernetics, 2012, 42(2): 482-500

[7] Bi Xiaojun, Liu Guo'an, Xiao Jing. Dynamic adaptive differential evolution based on novel mutation strategy [J]. Journal of Computer Research and Development, 2012, 49(6): 1288-1297 (in Chinese)
(毕晓君, 刘国安, 肖婧. 基于新变异策略的动态自适应差分进化算法[J]. 计算机研究与发展, 2012, 49(6): 1288-1297)

[8] Das S, Abraham A, Chakraborty U, et al. Differential evolution using a neighborhood based mutation operator [J]. IEEE Trans on Evolutionary Computation, 2009, 13(3): 526-553

[9] Chen Jie, Xin Bin, Peng Zhihong. Statistical learning makes the hybridization of particle swarm and differential evolution more efficient—A novel hybrid optimizer [J]. Science China: Information Sciences, 2009, 52(7): 1278-1282

[10] Das S, Konar A, Chakraborty U. Annealed differential evolution [C] //Proc of IEEE Congress Evolutionary Computation (CEC'2007). Piscataway, NJ: IEEE, 2007: 1926-1933

[11] Li Xiangtao, Yin Minghao, Ma Zhiqiang. Hybrid differential evolution and gravitation search algorithm for unconstrained optimization [J]. International Journal of Physical Sciences, 2011, 6(25): 5961-5981

[12] Rashedi E, Nezamabadi H, Saryazdi S. GSA: A gravitational search algorithm [J]. Information Sciences, 2009, 179(13): 2232-2248

[13] Xin Bin, Chen Jie, Peng Zhihong. An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization [J]. Science China: Information Sciences, 2010, 53(5): 980-989

[14] 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

Zhang Yingjie, born in 1970. PhD and associate professor. His main research interests include evolutionary computation and intelligent control.

Gong Zhonghan, born in 1988. Master. His main research interests include evolutionary computation and data mining (hncl18@hnu.edu.cn).