DOI: 10.13718/j.cnki.xdzk.2015.03.016

动态局部搜索差分进化算法①

张　伟，　刘三阳

西安电子科技大学 数学与统计学院，西安 710126

摘要：针对差分进化算法存在进化后期收敛速度慢、易早熟等缺点，提出了一种基于动态局部搜索的差分进化算法(DLSDE). 采用随机选择的方式进行变异并运用小概率扰动操作，增加种群的多样性，平衡算法的开发能力和探索能力；同时，对当前的最优解进行动态局部搜索，以加快算法的收敛速度. 对标准测试函数进行仿真实验并与其他 6 种算法进行比较，结果表明 DLSDE 算法具有较快的收敛速度和较高的求解精度，对复杂的数值优化问题寻优效果很好.

关　键　词：差分进化算法；随机选择；变异；扰动；动态局部搜索

中图分类号：TP18　　**文献标志码**：A　　**文章编号**：1673-9868(2015)03-0093-06

差分进化算法(differential evolution，DE)[1]是由 Rainer Storn 和 Kenneth Price 在 1995 年为求解切比雪夫多项式而提出的一种群智能优化算法. DE 算法借助于种群个体之间的差分信息对个体形成扰动来探索整个种群空间，并利用贪婪竞争机制选择下一代个体，寻求问题的最优解. DE 算法是一种基于实数编码的全局优化算法，有较强的全局搜索能力和收敛速度. DE 算法的主要特点是：算法简单，收敛速度快，所需领域知识少，适于解决比较复杂的优化问题.

目前，针对 DE 算法的相关研究已经取得了不少进展. 文献[2]提出自适应二次差分变异算法；文献[3]提出了基于三角法的变异算子，以提高 DE 算法的局部搜索能力；文献[4]提出了模糊自适应差分进化算法，基于模糊逻辑控制器自适应调整算法参数；文献[5]提出了中心变异差分进化算法；文献[6]提出了参数适应性分布差分进化算法，将种群分割并进行周期性交流，同时提出自适应交叉和变异策略，算法的收敛速度和解的性能有了较大的提高；文献[7]提出了 DEGL 算法，利用当前个体的临近个体的优势解以及其它临近个体关联的变异率，动态产生 F，用于个体的变异.

为了加快算法的收敛速度和提高寻优精度，本文受文献[8]的启发，在 DE 算法的迭代过程中加入动态局部搜索算子. 另外，对变异操作加以改动，采用随机选择[9]的方式进行变异和扰动操作，增加种群的多样性，平衡算法的开发能力和探索能力，避免算法陷入局部最优而导致早熟收敛. 数值实验表明改进的算法能显著提高优化性能.

1　基本差分进化算法

DE 算法是一种基于群体的进化算法，它与其他进化算法如遗传算法(GA)、进化规划(EP)、进化策略(ES)及其变种不同. DE 算法首先在解的取值范围内生成一个随机的初始种群，然后经过差分变异、交叉、选择操作，保存优秀个体，生成下一代种群，如此反复迭代，不断进化.

1.1　种群初始化

设种群规模为 NP，进化代数 $G=0,1,\cdots,G_{\max}$，种群中第 i 个个体表示为

① 收稿日期：2014-04-21
基金项目：国家自然科学基金项目(11301408).
作者简介：张　伟(1989-)，男，山西运城人，硕士研究生，主要从事进化计算、最优化理论与方法研究.

$$\boldsymbol{X}_{i,G}=(x_{1,i,G},\ x_{2,i,G},\ \cdots,\ x_{D,i,G}) \tag{1}$$

式中 D 表示个体变量的维数，则随机产生的初始种群为

$$x_{j,i,0}=x_{j,\min}+\text{rand}*(x_{j,\max}-x_{j,\min}) \tag{2}$$

式中：$x_{j,\max}$ 和 $x_{j,\min}$ 分别为个体变量 $\boldsymbol{X}_{i,G}$ 第 j 维的上界和下界，$j=1,2,\cdots,D$；rand 为[0，1]之间均匀分布的随机数.

1.2 变 异

对每一代进化目标向量 $\boldsymbol{X}_{i,G}$ 的变异操作如下：

$$\boldsymbol{V}_{i,G}=\boldsymbol{X}_{r1,G}+F(\boldsymbol{X}_{r2,G}-\boldsymbol{X}_{r3,G}) \tag{3}$$

其中：$r_1,r_2,r_3\in\{1,2,\cdots,NP\}$ 为互不相同，且与目标向量序号 i 也不同的随机数；F 为变异率.

1.3 交 叉

对目标向量 $\boldsymbol{X}_{i,G}$ 和它的变异向量 $\boldsymbol{V}_{i,G}$ 进行交叉，产生试验向量 $\boldsymbol{U}_{i,G}=(u_{1,i,G},\ u_{2,i,G},\ \cdots,\ u_{D,i,G})$，具体操作如下：

$$u_{i,j,G}=\begin{cases}v_{i,j,G},\ \text{rand}<CR\ \text{or}\ j=r\\ x_{i,j,G},\ \text{otherwise}\end{cases} \tag{4}$$

1.4 选 择

对试验向量和当前种群中的目标向量进行贪婪选择，具体方法如下：

$$\boldsymbol{X}_{i,G+1}=\begin{cases}\boldsymbol{U}_{i,G},\ f(\boldsymbol{U}_{i,G})\leqslant f(\boldsymbol{X}_{i,G})\\ \boldsymbol{X}_{i,G},\ \text{otherwise}\end{cases} \tag{5}$$

2 基于动态局部搜索的差分进化算法

2.1 随机变异操作

变异操作是差分进化一个非常重要的操作，DE 算法随着进化代数的增加，个体之间的差异性逐渐变小，使得算法早熟收敛. 为了避免算法早熟收敛，增加种群的多样性，本文采用文献[9]中的随机变异方法，具体表达式如下：

$$\boldsymbol{V}_{i,G+1}=\begin{cases}\boldsymbol{X}_{r0,G}+\text{rand}_1*(\boldsymbol{X}_{r1,G}-\boldsymbol{X}_{r2,G}),\ \text{if rand}>0.5\\ \boldsymbol{X}_{\text{best},G}+\text{rand}_2*(\boldsymbol{X}_{r1,G}-\boldsymbol{X}_{r2,G}),\ \text{if rand}\leqslant 0.5\end{cases} \tag{6}$$

其中：rand_1，rand_2，rand 都是[0，1]之间均匀分布的随机数；$\boldsymbol{X}_{r0,G}$，$\boldsymbol{X}_{r1,G}$，$\boldsymbol{X}_{r2,G}$ 是从当前种群中随机选取的个体，并且 r_0，r_1，r_2 互不相同；$\boldsymbol{X}_{\text{best},G}$ 为当前种群的最优个体. 随机变异操作能有效扩大算法的搜索性能，且能兼顾算法的探索能力和开发能力，可以取得较好的寻优效果.

2.2 扰动操作

DE 算法经过变异、交叉和选择操作后，使得当前的解向最优解靠近，最优解可能是全局最优，也可能是局部最优. 如果是局部最优解，就有可能使种群陷入局部最优. 为了避免这种情况本文采用一种小概率扰动策略，使得当前的解能够跳出局部最优，具体表达式如下：

if rand $<$ Mr

$$\boldsymbol{V}_{i,G+1}=\begin{cases}\boldsymbol{X}_{r0,G}+\text{rand}_1*(\boldsymbol{X}_{r1,G}-\boldsymbol{X}_{r2,G}),\ \text{if rand}>0.5\\ \boldsymbol{X}_{\text{best},G}+\text{rand}_2*(\boldsymbol{X}_{r1,G}-\boldsymbol{X}_{r2,G}),\ \text{if rand}\leqslant 0.5\end{cases} \tag{7}$$

else

$$\boldsymbol{V}_{i,G+1}=\boldsymbol{X}_{\min}+\text{rand}*(\boldsymbol{X}_{\max}-\text{x}_{min})$$

end

其中：$\boldsymbol{X}_{\min}$ 和 $\boldsymbol{X}_{\max}$ 分别为搜索区间的上下界，Mr 一般取大于 0.9 的值.

2.3 动态局部搜索

动态搜索技术[10]是在基本随机搜索技术的基础上提出来的，它由一般搜索和局部搜索构成. 它主要是在当前解的邻域内进行随机搜索，并且搜索步长随着迭代次数逐渐缩小(缩小比例为 0.5)，使得当前解及其周围邻域得到充分搜索. 标准的 DE 算法缺乏局部搜索能力，导致算法在进化后期收敛速度变慢. 为了弥

补该缺点，本文将动态局部搜索算子嵌入到DE算法中，利用其强大的开发能力，对当前的最优解进行多次局部搜索，使其快速收敛到搜索区域的最优解. 动态局部搜索算子的具体方法如下：

算法 1　（动态局部搜索算子）

1）参数初始化，局部搜索迭代次数 E，初始搜索步长 $\boldsymbol{\alpha}_0$，局部搜索迭代计数器 epoch$=0$，$k=0$，$\boldsymbol{X}_{\text{current}}=\boldsymbol{X}_{\text{best}}$.

2）迭代计数器 $n=0$.

3）生成随机向量 $\mathrm{d}\boldsymbol{X}$，且满足 $-\boldsymbol{\alpha}_k \leqslant \mathrm{d}\boldsymbol{X} \leqslant \boldsymbol{\alpha}_k$.

4）更新 epoch，epoch$=$epoch$+1$.

5）$f_{\text{new}}=f(\boldsymbol{X}_{\text{current}}+\mathrm{d}\boldsymbol{X})$，

若 $f_{\text{new}}<f_{\text{best}}$，则 $f_{\text{best}}=f_{\text{new}}$，$\boldsymbol{X}_{\text{best}}=\boldsymbol{X}_{\text{current}}+\mathrm{d}\boldsymbol{X}$，$n=n+1$，转 7）；

若 $f_{\text{new}}<f_{\text{current}}$，则 $f_{\text{current}}=f_{\text{new}}$，$\boldsymbol{X}_{\text{current}}=\boldsymbol{X}_{\text{current}}+\mathrm{d}\boldsymbol{X}$，$n=n+1$，转 7）.

6）$f_{\text{new}}=f(\boldsymbol{X}_{\text{current}}-\mathrm{d}\boldsymbol{X})$，

若 $f_{\text{new}}<f_{\text{best}}$，则 $f_{\text{best}}=f_{\text{new}}$，$\boldsymbol{X}_{\text{best}}=\boldsymbol{X}_{\text{current}}-\mathrm{d}\boldsymbol{X}$，$n=n+1$，转 7）；

若 $f_{\text{new}}<f_{\text{current}}$，则 $f_{\text{current}}=f_{\text{new}}$，$\boldsymbol{X}_{\text{current}}=\boldsymbol{X}_{\text{current}}-\mathrm{d}\boldsymbol{X}$，$n=n+1$，转 7）.

7）如果 $n<N$，则转 3）.

8）$k=k+1$.

9）$\boldsymbol{\alpha}_k=\boldsymbol{\alpha}_{k-1}*0.5$.

10）如果 epoch$=E$，则算法终止；否则转 2）.

其中，$\boldsymbol{X}_{\text{best}}$ 为当前的最优解，f_{best}，f_{new}，f_{current} 分别为对应解的适应值.

为了充分发挥动态局部搜索算子的开发能力，初始搜索步长 $\boldsymbol{\alpha}_0$ 应与当前最优解的数量级尽量保持一致，本文取 $\boldsymbol{\alpha}_0=\boldsymbol{X}_{\text{best}}$.

2.4　基于动态局部搜索的差分进化算法

基于动态局部搜索的差分进化算法的基本思路是：采用随机选择的方式进行变异并进行小概率扰动操作，增加种群的多样性，平衡算法的开发能力和探索能力；同时，对当前的最优解进行动态局部搜索，以加快算法的收敛速度，使算法具有很好的寻优效果.

算法 2　基于动态局部搜索的差分进化算法

1）初始化种群规模 SN 与函数评价次数 Max. FE；

2）随机初始化种群并计算适应值；

3）若 FE$<$Max. FE，转 4），否则算法终止；

4）for $i=1$ to SN

5）对个体 $\boldsymbol{X}_i$ 按照(7) 式进行变异操作，产生变异个体 $\boldsymbol{V}_i$；

6）对个体 $\boldsymbol{X}_i$ 和它的变异个体 $\boldsymbol{V}_i$ 按照(4) 式进行交叉操作产生实验个体 $\boldsymbol{U}_i$；

7）按照(5) 式贪婪选择个体 $\boldsymbol{X}_i$ 与个体 $\boldsymbol{U}_i$；

8）end for

9）找出当前的最优解，并按照算法 1 对最优解进行动态局部搜索；

10）FE$=$FE$+1$，转 3）.

3　实验仿真与结果分析

为了测试本文提出的基于动态局部搜索的差分进化算法的性能，我们选取了 5 个 30 维的 Benchmark 函数作为测试函数对本文的算法进行测试，并选取经典的 DE/rand/1 算法、DE/best/1 算法、PSO 算法、SaDE 算法、JADE 算法、RMDE 算法[9] 与本文的算法进行比较. 表 1 给出了 5 个测试函数的表达式、搜索区间范围和最优值. 数值实验在 Inter Xeon，CPU E5620，2.40 GHz 主频的计算机上进行，程序采用 Matlab2013a 语言实现. 算法的参数设计如下：DE/rand/1 算法和 DE/best/1 算法中 F$=0.5$，CR$=0.9$；PSO 算法、SaDE 算法及 JADE 算法参数根据文献[11－12] 设置，RMDE 算法的参数根据文献[9] 设置，本

文的参数设置为 $F=0.5$，$CR=0.9$，$E=30$，$N=3$. 所有的种群规模都取100. 表1统计了在相同的条件下，各种算法独立实验30次的最优值、最差值、平均值和标准差. 其中，最优值、最差值、平均值反映了解的质量，标准差反映了算法的稳定性和鲁棒性，结果如表2所示.

表1 测试函数

函数	函数名	搜索区间	最小值
$f_1(X)=\sum_{i=1}^{D}x_i^2$	Sphere	$[-100,100]^D$	0
$f_2(X)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	Rastrigin	$[-5.12,5.12]^D$	0
$f_3(X)=\sum_{i=1}^{D-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	Rosenbrock	$[-30,30]^D$	0
$f_4(X)=-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2})-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)+20+e$	Ackley	$[-32,32]^D$	0
$f_5(X)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	Schwefel 2.2	$[-10,10]^D$	0

表2给出了6种算法对5个测试函数结果性能的比较. 可以看出，对于Sphere函数，在其他5种算法函数评价次数为 1.5×10^5 而DLSDE算法函数评价次数为 2.5×10^4 的情况下，DLSDE算法的求解精度比其他5种算法高出很多；对于Rastrigin函数，DLSDE在函数评价次数为 2×10^3 时就达到最优值0；对于Rosenbrock函数除了Best项稍差于RMDE，其他项也取得了不错的效果；对于Ackley函数，DLSDE算法的求解精度也有所提高；对于Schwefel2.2函数，在函数评价次数为 2×10^5 时，DLSDE算法的求解也优于其他5种算法. 通过对比可以看出，DLSDE算法表现出了极高的收敛精度.

通过以上的分析可以看出，本文提出的DLSDE算法展示了很强的收敛精度，能够较大幅度提高测试函数求解精度，算法的收敛速度也有显著提高，鲁棒性较好，是一种很好的函数寻优方法.

为了更加直观地反映算法的寻优效果，将DLSDE算法与DE算法进行比较，给出两种算法的收敛曲线图. 从图1可以直观看出来，无论是收敛速度还是算法的精度，DLSDE算法较之DE算法有了很大提高.

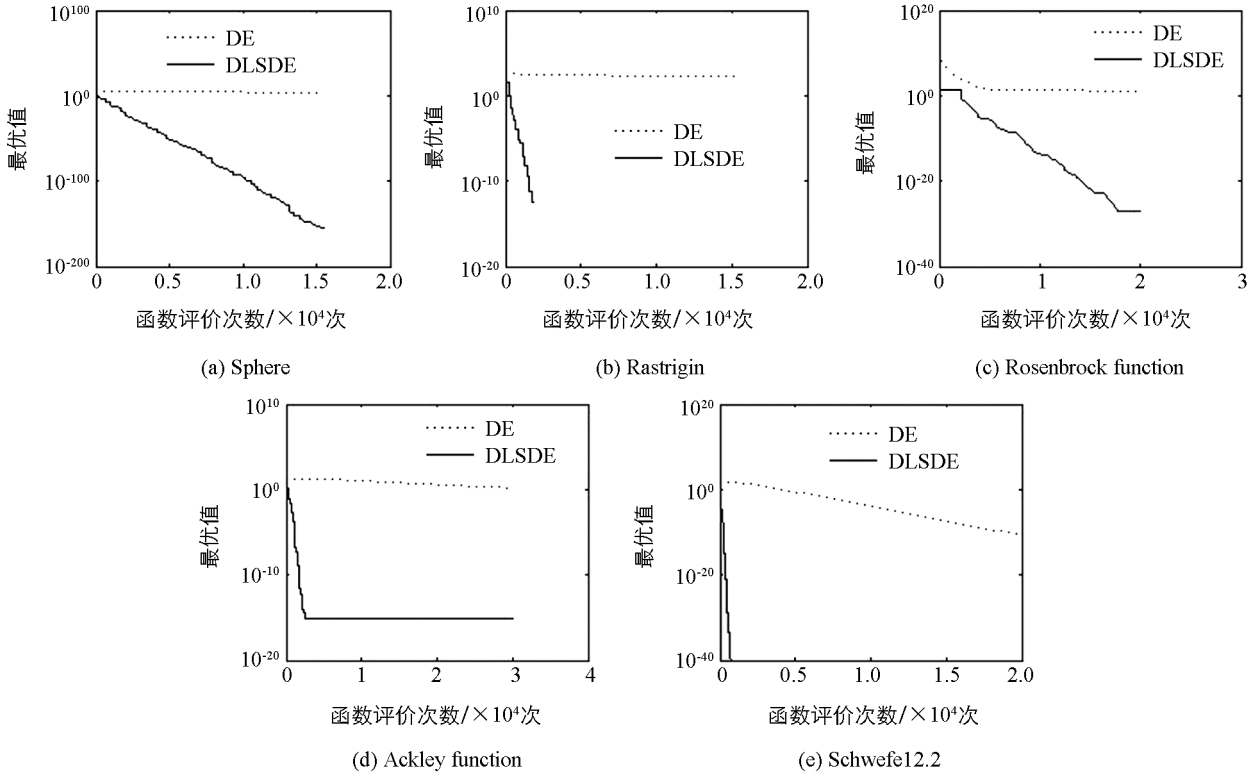

图1 5个函数的收敛性能比较

表 2　6 种算法对 5 个测试函数结果性能的比较

函数	函数评价次数	算法	最优值	最差值	平均值	标准差
f_1	1.5×10^5	DE/rand/1	8.25e-012	8.19e-009	1.39e-009	2.25e-009
		DE/best/1	2.27e-029	5.20e-027	8.56e-028	1.02e-027
		PSO	1.37e-041	1.81e-031	8.07e-033	3.35e-032
		SaDE	2.34e-046	3.05e-042	4.93e-044	1.17e-043
		JADE	9.37e-065	9.29e-059	8.02e-060	2.26e-059
		RMDE	5.28e-118	2.16e-105	7.26e-107	3.94e-106
	2.5×10^4	DLSDE	1.49e-310	5.77e-263	1.92e-264	0
f_2	3×10^5	DE/rand/1	8.96e＋000	3.07e＋001	1.70e＋001	2.84e＋001
		DE/best/1	3.31e＋000	2.74e＋001	1.48e＋001	2.07e＋001
		PSO	2.92e-012	2.92e-008	1.85e-011	1.09e-011
		SaDE	1.89e-013	1.13e-011	2.79e-012	4.05e-012
		JADE	3.10e-013	2.88e-012	8.99e-012	5.11e-012
		RMDE	7.38e-016	2.91e-014	6.79e-015	6.63e-015
	2×10^3	DLSDE	0	0	0	0
f_3	2×10^5	DE/rand/1	3.75e＋000	8.81e＋001	3.68e＋001	2.52e＋001
		DE/best/1	2.34e＋000	1.60e＋001	3.05e＋000	3.68e＋000
		PSO	1.11e＋000	2.83e＋001	2.04e＋000	4.12e＋000
		SaDE	4.62e-004	9.11e-003	9.71e-004	5.84e-004
		JADE	3.71e-022	4.89e-020	4.89e-021	9.39e-021
		RMDE	1.51e-028	2.46e-024	2.31e-026	4.80e-026
	2×10^5	DLSDE	7.36e-28	1.85e-26	8.88e-27	8.88e-27
f_4	2×10^5	DE/rand/1	1.59e-008	2.89e-007	3.59e-008	3.08e-008
		DE/best/1	3.89e＋001	5.14e＋001	3.11e＋001	2.87e＋001
		PSO	1.01e-008	3.11e-007	2.89e-008	3.84e-008
		SaDE	1.24e-013	4.10e-013	3.02e-013	2.36e-013
		JADE	8.59e-015	9.01e-014	7.98e-015	6.51e-015
		RMDE	3.86e-015	7.09e-015	4.51e-015	1.62e-015
	2×10^5	DLSDE	8.88e-16	8.88e-16	8.88e-16	8.88e-16
f_5	2×10^5	DE/rand/1	1.33e-019	3.10e-18	2.84e-018	3.88e-018
		DE/best/1	1.71e-009	2.09e-008	8.96e-009	1.06e-008
		PSO	1.94e-014	6.52e-012	4.08e-013	2.82e-013
		SaDE	9.35e-021	3.16e-020	7.97e-020	5.95e-020
		JADE	3.88e-035	8.09e-034	2.80e-034	3.92e-035
		RMDE	5.06e-043	3.10e-041	4.95e-042	7.83e-042
	2×10^5	DLSDE	8.03e-128	7.05e-107	2.42e-108	1.28e-107

4　结束语

为了加快DE算法的收敛速度并避免陷入局部最优，提出了一种基于动态局部搜索的差分进化算法. 通过对目前找到的最优解进行动态局部搜索，加快了算法的开发能力；同时，采用随机选择的方式进行变异并进行小概率扰动操作以增加种群的多样性. 对标准测试函数的仿真实验以及与其他算法的比较实验表明本文提出的算法具有较快的收敛速度和较高的求解精度，对复杂的数值优化问题寻优效果很好，进一步的研究可将算法推广到其他领域，如参数优化和线性系统逼近等优化问题.

参考文献:

[1] STORN R, PRICE K. Differential Evolution-A Simple And Efficient Adaptive Scheme For Global Optimization Over Continuous Spaces [J]. Journal of Global Optimization, 1997(11): 341－359.

[2] 吴亮红, 王耀南, 袁小芳, 等. 自适应二次变异差分进化算法 [J]. 控制与决策, 2006, 21(8): 898－902.

[3] FAN H Y, LAMPINEN J. A Trigonometric Mutation Operation to Differential Evolution [J]. Journal of Global Optimization, 2003, 27(1): 105－129.

[4] LIU J, LAMPINEN J. A Fuzzy Adaptive Differential Evolution Algorithm [J]. Soft Computing-a Fusion of Foundations. Methodologies and Applications, 2005, 9(6): 448－462.

[5] 池元成, 方　杰, 蔡国飙. 中心变异差分进化算法 [J]. 系统工程与电子技术, 2010, 32(5): 1105－1108.

[6] 张春梅, 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.

[7] DAS S, ABRAHAM A, CHAKRABORTY U K. Differential Evolution Using a Neighborhood-Based Mutation Operator [J]. IEEE Trans Evolut Comput, 2009, 13(3): 526－553.

[8] 刘三阳, 张　平, 朱明敏. 基于局部搜索的人工蜂群算法 [J]. 控制与决策, 2014, 29(1): 123－128.

[9] 欧阳海滨, 高立群, 孔祥勇. 随机变异差分进化算法 [J]. 东北大学学报: 自然科学版, 2013, 34(3): 330－334.

[10] HAMZACEBI C, KUTAY F. Continuous Functions Minimization by Dynamic Random Search Technique [J]. Applied mathematical Modelling, 2007, 31(10): 2189－2198.

[11] QIN A K, SUGANTHAN P N. Self-Adaptive Differential Evolution Algorithm for Numerical Optimization [C]//IEEE Congress on Evolution Computation. New York: IEEE Press, 2005: 1785－1791.

[12] ZHANG J Q, SANDERSON A C. Adaptive Differential Evolution With Optional External Archive [J]. IEEE Transactions on Evolutinary Computation, 2009, 13(5): 945－958.

Dynamic Local Search Differential Evolution Algorithm

ZHANG　Wei,　LIU San-yang

School of Mathematics and Statistics, Xi Dian University, Xi'an 710126, China

Abstract: Aiming at the shortcoming of differential evolution (DE), such as the low convergence rate in the late evolution and easy to be trapped into the local optimums, an improved DE algorithm based on local search is proposed in this paper. The random choice method and small probability perturbation are adopted to increase the diversity of the population and to balance exploitation and exploration of the algorithm. Full use is made of dynamic local search (DLS) to optimize the current best solution to speed up the convergence rate. Simulation experiments are conducted on a suite of benchmark functions and the results are compared with those of other six algorithms. The results demonstrate that the DLSDE algorithm has a faster convergence rate and higher solution accuracy and shows good performance in solving complex numerical optimization problems.

Key words: differential evolution algorithm; random choice; mutation; disturbance; dynamic local search

责任编辑　张　枸