基于动态自适应策略的改进差分进化算法

王丛佼 王锡淮 肖健梅
(上海海事大学电气自动化系 上海 201306)

摘 要 针对差分进化算法处理复杂优化问题时存在后期收敛速度变慢、收敛精度不高和参数设置困难的问题,提出了一种基于动态自适应策略的改进差分进化算法(*dn*-DADE)。首先,新的变异策略 DE/current-to-*dn*best/1 利用当前种群中的精英解引导有效的搜索方向来动态调整可选的精英解,使其在进化后期趋于全局最优解。其次,分别设计了缩放因子和交叉因子的自适应更新策略,使两者在搜索的不同阶段自适应变化,以弥补差分进化算法对参数敏感的不足,进一步提高算法的稳定性和鲁棒性。对 14 个 benchmark 函数进行了测试并与多种先进 DE 改进算法进行了比较,结果显示,*dn*-DADE 算法具有较高的求解精度,收敛速度快,寻优性能显著。

关键词 差分进化,变异策略,动态调整,参数自适应,全局优化
中图法分类号 TP18 **文献标识码** A

Improved Differential Evolution Algorithm Based on Dynamic Adaptive Strategies

WANG Cong-jiao WANG Xi-huai XIAO Jian-mei
(Department of Electrical Engineering and Automation, Shanghai Maritime University, Shanghai 201306, China)

Abstract To solve problems of DE applied to complex optimization functions, an improved differential evolution algorithm (*dn*-DADE) based on dynamic adaptive strategy was proposed in this paper. Firstly, the elite solutions of current population were utilized in the new mutation strategy (DE/current-to-*dn*best/1) to guide the search direction. Secondly, the adaptive update strategies of scaling factor and crossover factor were designed for making parameter values self-adapting at different search stages to improve the stability and robustness of the algorithm. A set of 14 benchmark functions were adopted to test the performance of the proposed algorithm. The results show that *dn*-DADE algorithm has the advantages of remarkable optimizing ability, higher search precision, faster convergence speed and outperforms several state-of-the-art improved differential evolution algorithms in terms of the main performance indexes.

Keywords Differential evolution, Mutation strategy, Dynamic adjustment, Parameter self-adaptation, Global optimization

1 引言

差分进化(Differential Evolution, DE)算法[1]是一种通过引入独特的差分变异算子进行迭代搜索的全局优化算法。DE 算法作为目前最前沿、最具代表性的进化算法之一,现已被成功地应用于各类科学研究与实际工程领域[2-4]中。但 DE 算法在实际应用中仍存在一些不足[5,6],如求解难度较大的高维多峰复杂优化问题时,易出现后期收敛速度慢、局部搜索能力不足和陷入局部最优等问题。

研究发现 DE 算法的性能高度依赖其变异交叉策略和相应参数(缩放因子 F、交叉因子 CR 和种群大小 NP)的设置[7,8],针对不同的具体问题,不合适的变异交叉策略和参数选择很可能会造成算法的早熟收敛或停滞。目前,DE 算法有多种可选的变异策略,选取何种变异策略来平衡算法的探测能力和开采能力是很重要的[9]。同时,为改善 DE 算法的性能,DE 研究者主要针对参数 F 和 CR 设计了不同的自适应机制。文献[7]的 SaDE 算法提出 F 采用随机正态分布产生,CR 则利用进化过程中的反馈信息进行自适应调整,使算法取得了良好的优化效果。文献[10]提出了一种自适应的 SDE 算法,该算法中 F 通过类似差分进化算法的变异算子自适应控制。文献[11]提出的 JADE 算法中,每个个体对应一个服从柯西分布的 F 和一个服从正态分布的 CR,然后采用联系更新的方式自适应地调整两种分布的参数。

为了进一步提高 DE 算法的全局收敛性,弥补其后期收敛速度较慢和对参数设置敏感的不足,本文提出了一种新的变异策略 DE/current-to-*dn*best/1,并分别对 F 和 CR 设计了新的自适应调整机制,构成了一种综合改进的基于动态自适应策略的差分进化算法(*dn*best-based Dynamic Adaptive Differential Evolution, *dn*-DADE)。不同于 DE/current-to-best/1 始终利用全局最优解来引导种群进化方向,DE/current-to-*dn*best/1 通过当前种群中的精英解在相对较大的范围内搜索,在进化前期增强算法的探索能力,同时动态变化可选的局

到稿日期:2013-02-03 返修日期:2013-05-25 本文受上海市教委科研创新重点项目(12ZZ158),上海市教委重点学科建设项目(J50602)资助。
王丛佼(1984—),女,博士生,主要研究方向为智能控制、智能信息处理等;王锡淮(1961—),男,教授,博士生导师,主要研究方向为智能控制、智能信息处理、智能图像处理等;肖健梅(1962—),女,教授,主要研究方向为智能控制、智能信息处理等。

部精英解的个数，使其在进化后期愈趋近于全局最优解，加快后期收敛速度，从而平衡算法的全局搜索和局部搜索；另一方面，在仔细研究 F 和 CR 两个参数的基础上，分别提出了两者的自适应调整方案，以增强算法的鲁棒性。通过对 14 个 CEC2005 竞赛提供的 Benchmark 函数的性能测试，以及与现有先进的 DE 改进算法和 *dn*-DADE 算法自身变种的比较，表明了该算法的有效性和高效性。

2 基本差分进化算法

DE 算法是采用实数编码的进化算法，具有与其他进化算法相似的遗传操作，即通过对个体进行变异、交叉和选择等操作，使种群不断进化直至算法结束。

2.1 种群初始化

假设种群规模为 NP，即有 NP 个个体参与进化，进化代数 $G=0,1,\cdots,G_{\max}$。则种群中第 i 个个体表示为：

$$X_{i,G}=[x_{1,i,G},x_{2,i,G},x_{3,i,G},\cdots,x_{D,i,G}] \tag{1}$$

式中，D 表示个体变量的维数。则初始种群(即 $G=0$)中每个个体在目标函数指定的$[X_{\min},X_{\max}]$值域内随机均匀产生：

$$x_{j,i,0}=x_{j,\min}+rand_{i,j}[0,1]\cdot(x_{j,\max}-x_{j,\min}) \tag{2}$$

式中，$x_{j,\max}$和 $x_{j,\min}$分别为个体变量 $X_{i,G}$第 j 维的上界和下界，$j=1,2,\cdots,D$；$rand_{i,j}[0,1]$是$[0,1]$区间内均匀分布的随机数。

2.2 变异操作

DE 算法利用种群中不同个体间的差分向量对个体进行扰动，产生变异向量 $V_{i,G}$。通常采用 $DE/a/b$ 来表示不同的变异策略，其中 a 代表基向量的选择方式，b 表示差分向量的个数。目前比较常用的变异策略有：

DE/rand/1：

$$V_{i,G}=X_{r1,G}+F\cdot(X_{r2,G}-X_{r3,G}) \tag{3}$$

DE/best/2：

$$V_{i,G}=X_{best,G}+F\cdot(X_{r1,G}-X_{r2,G})+F\cdot(X_{r3,G}-X_{r4,G}) \tag{4}$$

DE/current-to-best/1：

$$V_{i,G}=X_{i,G}+F\cdot(X_{best,G}-X_{i,G})+F\cdot(X_{r1,G}-X_{r2,G}) \tag{5}$$

式中，$r_1,r_2,r_3,r_4\in[1,2,\cdots,NP]$，代表种群中不同个体的索引号，是不等于 i 且互不相同的随机整数；$X_{best,G}$为 G 代种群中适应度最好的个体。缩放因子 $F\in(0,2)$，对差分向量进行缩放。

2.3 交叉操作

DE 算法有两种交叉方式：二项式交叉和指数交叉。目标向量 $X_{i,G}$与变异向量 $V_{i,G}$进行离散交叉，从而产生试验向量 $U_{i,G}=[u_{1,i,G},u_{2,i,G},\cdots,u_{D,i,G}]$。本文采用的二项式交叉操作如式(6)所示：

$$u_{j,i,G}=\begin{cases}v_{j,i,G}, rand(j)\leqslant CR \text{ or } j=randn(i)\\ x_{j,i,G}, rand(j)>CR \text{ and } j\neq randn(i)\end{cases} \tag{6}$$

式中，$rand(j)\in[0,1]$为均匀分布的随机数；$randn(i)\in[1,2,\ldots,D]$为随机选择的维数变量索引，用以保证 $U_{i,G}$至少有一位由 $V_{i,G}$贡献，而对于其它位由交叉概率因子 CR 决定，$CR\in[0,1]$。

2.4 选择操作

DE 算法采用贪婪选择策略，即只有适应度函数值更优的个体进入到下一代。对于求解全局最小化问题，选择操作表示为：

$$X_{i,G+1}=\begin{cases}U_{i,G}, f(U_{i,G})<f(X_{i,G})\\ X_{i,G}, f(U_{i,G})\geqslant f(X_{i,G})\end{cases} \tag{7}$$

式中，$X_{i,G+1}$为下一代的第 i 个个体，f 为适应度函数。

3 基于动态自适应策略的改进差分进化算法

本文提出的 *dn*-DADE 算法从变异策略和参数自适应出发，对 DE 算法进行了综合改进。

3.1 新的变异策略 DE/current-to-*dn*best/1

DE/rand/1 是最早被采用的变异策略[1]，在 DE 算法的研究中得到了广泛和成功的应用。随后文献[12]指出，DE/best/2 和 DE/current-to-best/1 也具有很大的性能优势。相比 DE/rand/k，更具贪婪性的 DE/best/k 和 DE/current-to-best/k 由于利用最优解来引导种群进化，因此具有更快的收敛速度，全局搜索能力更强。但其过于强调算法的开采能力，个体在当前最优解的周围过度聚集，可能导致算法陷入局部最优解。

为此，本文在 DE/current-to-best/1 的基础上提出了新的 DE/current-to-*dn*best/1 策略。该策略的思想是，先将当前种群的个体根据适应度函数值进行排序，对于全局最小化问题，适应度函数值越小的个体排名越前，将前 $dn(dn\in[1,2,\cdots,NP])$个个体定义为当前种群精英解。算法利用随机选取的种群精英解来引导搜索方向，同时 dn 随迭代次数非线性递减，这样可以在进化前期兼顾种群多样性，而在算法后期使精英解的选择趋于当前最优解，从而加快收敛速度。其具体形式如下：

$$V_{i,G}=X_{i,G}+F_i\cdot(X_{dnbest,G}-X_{i,G})+F\cdot(X_{r_1^i,G}-X_{r_2^i,G}) \tag{8}$$

式中，$X_{dnbest,G}$表示当前群体前 dn 个精英解中的任意一个。$X_{r_1^i,G}$和 $X_{r_2^i,G}$是从当前种群中随机选取的不同于 $X_{dnbest,G}$或 $X_{i,G}$的个体。dn 的函数表达式为：

$$dn=ceil\left\{\frac{NP}{4}\cdot\left[\cos\left(\frac{G}{G_{\max}}\cdot\pi\right)+1\right]\right\} \tag{9}$$

式中，G 为当前进化代数，$G_{\max}$为最大进化代数。$ceil(y)$表示大于 y 的最小整数。如图 1 所示(以 $G_{\max}=5000$ 为例)，随着进化代数的增加，dn 从 $NP/2$ 缓慢减少到 1。dn 在搜索前期其下降速率较慢，算法在相对较大的范围内寻求可行解，增强了算法的局部搜索能力；而在搜索后期下降速率相对较快，充分利用相对优秀解所提供的精英解分布区域的信息指导搜索，当 $dn=1$ 时，$X_{dnbest,G}=X_{best,G}$为当前最优解，算法更倾向在全局最优解附近搜索，从而加快了收敛速度。由此可见，DE/current-to-*dn*best/1 策略能够更好地平衡算法的探索能力和开采能力。

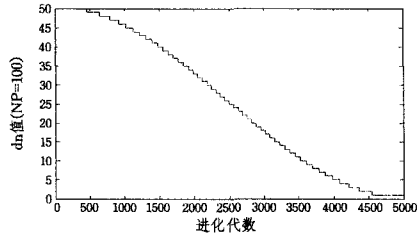


图 1 $G_{\max}=5000$ 时 dn 的变化曲线

3.2 参数动态自适应策略

传统 DE 算法对参数的选择非常敏感，参数设置会影响算法的收敛性能和寻优效率。其中交叉概率因子 CR 往往与所求问题的性质相关；缩放因子 F 则对收敛速度影响较大。

本文提出的 dn-DADE 算法对 CR 和 F 进行动态自适应调整，其基本思想是：采用正态分布生成 CR，并通过新的调整规则来利用使试验个体成功进入下一代的 CR 引导产生新的 CR 值；对于 F，则选取柯西算子作为其生成算子，同时考虑进化初期 F 值应较大，以保持种群多样性，增大搜索到全局最优解的概率，而在进化后期 F 值应趋小，使个体向全局最优解快速靠拢，促进种群收敛，从而通过调整柯西分布的位置参数来实现。

3.2.1 自适应交叉概率因子

本文提出的 dn-DADE 算法利用正态分布随机数自适应调节 CR，并受文献[7]中 SaDE 算法通过进化过程中的反馈信息更新概率分布的思想启发，对正态分布的均值和方差进行适应性调整。与 SaDE 算法不同，本文采用一种考虑个体适应度的加权的更新策略。

算法中每个个体的交叉概率因子 $CR_{i,G}$ 表示为：

$$CR_{i,G}=N(CR_{dn},\sigma_{dn}^2) \tag{10}$$

式中，$N(CR_{dn},\sigma_{dn}^2)$ 表示均值为 CR_{dn}、方差为 σ_{dn}^2 的正态分布随机函数。当 $CR_{i,G}$ 的值超出[0,1]范围时，则采用截断的方式使其在[0,1]间。在每一代进化过程中，试验个体成功进入下一代时所对应的 CR 值被保存在一个数组 M_{CR} 中，同时在数组 $M_{\Delta f}$ 中保存成功进化个体的适应值修正度，其定义公式如下：

$$\Delta f(k)=\frac{f(k)-f_{new}(k)}{f(k)} \tag{11}$$

式中，$f(k)$ 和 $f_{new}(k)$ 分别表示个体进化前后的适应度值。则每一个成功进化个体的交叉概率因子的贡献度可表示为：

$$\omega_k=\frac{\Delta f(k)}{\sum_{k=1}^{|M_{\Delta f}|}\Delta f(k)} \tag{12}$$

正态分布的均值 CR_{dn} 初始设为 0.5，之后每一代所有个体的 CR_{dn} 值按式(13)更新：

$$CR_{dn}=\sum_{k=1}^{|M_{CR}|}\omega_k\times M_{CR}(k) \tag{13}$$

式中，$|M_{\Delta f}|=|M_{CR}|$，即为成功进入下一代的试验个体的个数。对于正态分布的方差 σ_{dn}^2，同样采用自适应方法对其进行调整：

$$\sigma_{dn}^2=\frac{\sum_{k=1}^{|M_{CR}|}(M_{CR}(k)-CR_{dn})^2}{|M_{CR}|} \tag{14}$$

3.2.2 自适应缩放因子

与正态分布相比，柯西分布其两翼分布更广的特性不仅可以提供多种步长选择，并且能增加搜索步长的广度，从而扩大寻优范围，帮助算法摆脱局部极值点的干扰，因此本文 dn-DADE 算法中第 G 代个体的缩放因子 $F_{i,G}$ 按式(15)独立产生：

$$F_{i,G}=C(F_{dn},r) \tag{15}$$

式中，$C(F_{dn},r)$ 表示按照位置参数 F_{dn} 和缩放参数 r 产生的柯西分布随机数。由于较小的缩放因子 F 值容易使算法局部收敛，而较大的 F 值会降低收敛速度，甚至导致算法难以收敛，因此将实际产生的 $F_{i,G}$ 值控制在其截断区间[$F_{\min}$，$F_{\max}$]（$F_{\min}$ 取 0.2，$F_{\max}$ 取 0.8）。本文设置 $r=0.05$，F_{dn} 的初始值取 0.7，之后每一代根据进化代数动态更新：

$$\begin{cases}F_{dn}=F_{\max}'-(F_{\max}'-F_{\min}')\times(\dfrac{G}{G_{\max}})^{1/2}\\ F_{\max}'=F_{\max}-\theta r\\ F_{\min}'=F_{\min}+\theta r\end{cases} \tag{16}$$

根据柯西分布的特点，F_{dn} 落在区间[$F_{\min}$，$F_{\max}$]的边界附近会导致 $F_{i,G}$ 的取值有较大的概率超出其截断区间，因此 F_{dn} 的取值区间需要添加一个调整量 $\theta r(\theta=1,2,3,\cdots)$，$\theta$ 的作用和相关取值将在实验分析中说明。

4 算法测试与分析

4.1 测试函数

为验证 dn-DADE 算法的有效性，采用 14 个 CEC2005 国际竞赛所提供的标准函数进行测试。这些函数基于经典的 benchmark 函数（如 Schwefel、Rosenbrock、Rastrigin 和 Ackley 函数等），涵盖了自变量独立、自变量相互依赖和带噪音等不同特征，由于具有偏移旋转性，因此求解难度较大。具体函数见文献[13]，其中 f_1-f_5 为单峰函数，f_6-f_{14} 为多峰函数（f_{13} 和 f_{14} 是由基础函数组合而成的混合函数）。

4.2 实验设置

在本文实验中，首先将 dn-DADE 算法与 4 种目前先进的 DE 改进算法（包括 jDE[8] 算法、SaDE[7] 算法、JADE[11] 算法和 DEahcSPX[14] 算法）进行对比计算。其中比较算法均使用原文献推荐的参数设置：jDE 算法中 $F_l=0.1$，$F_u=0.9$，$\tau_1=\tau_2=0.1$；JADE 算法带归档，$c=0.1$，$p=0.05$；SaDE 算法参数见文献[7]；DEahcSPX 算法中 $F=0.9$，$CR=0.9$，$np=3$。

随后，为了进一步说明 dn-DADE 算法中参数自适应和新变异策略的作用，设计其本身的变种 dn-DE 算法、DADE 算法及 dnc-DADE 算法与 dn-DADE 算法进行比较，相关算法具体见 4.3 节。为保证比较的公平性，所有算法对各函数都独立运行 50 次，无论维数高低统一种群规模 $NP=100$，且每次独立运行都采用相同的初始种群。实验均在硬件配置为 Intel Core™ i5 CPU 2.5GHz、4GB 内存的机器上完成，程序采用 MATLAB7.0 编写。

4.3 与先进的优化算法的比较分析

为验证 dn-DADE 算法的整体性能优势，将其与 jDE 算法、SaDE 算法、JADE 算法和 DEahcSPX 算法分别在 30 维（$D=30$）和 100 维（$D=100$）的函数 f_1-f_{14} 上进行测试，比较各算法在最终解的质量、收敛速度和鲁棒性上的性能差别。由于中心偏移旋转函数的最优值不在原点，本文采用实际输出结果与理论最优结果之间的误差值作为评价准则。

设 x^* 为目标函数的已知全局最优解，x 为算法实际找到的最优解，定义误差值 EV（Error Value）为：$EV=f(x)-f(x^*)$。记录每次运行达到最大函数评价次数 MAX_FES 时（$D=30$ 时 $MAX\_FES=3\times10^5$；$D=100$ 时 $MAX\_FES=1\times10^6$）的 EV 最小值，统计 50 次运行结果的平均值和标准差，则表 1 和表 2 分别为 5 种算法在函数 $D=30$ 和 $D=100$ 时运行的统计结果，黑体部分表示所有算法在同一测试函数上得到的最优结果。

同时，本文使用 Wilcoxon 秩和检验[15]（显著度水平 5%）对表 1 和表 2 的数据进行了统计分析，Wilcoxon 秩和检验的 P 值分别见表 3 和表 4。

表 1　5 种算法在测试函数 f_1-f_{14} 上的 EV 平均值和标准差比较（$D=30$）

函数 \ 算法	jDE	SaDE	JADE	DEahcSPX	dn-DADE
	平均值（标准差）	平均值（标准差）	平均值（标准差）	平均值（标准差）	平均值（标准差）
$f_1(x)$	2.18E−25（6.92E−27）	1.09E−27（6.42E−28）	7.75E−55（6.48E−54）	2.77E−24（5.34E−25）	**7.25E−58（1.83E−60）**
$f_2(x)$	3.65E−08（7.41E−09）	6.27E−09（7.29E−10）	4.16E−25（8.57E−26）	8.14E−06（1.31E−05）	**5.17E−26（4.64E−26）**
$f_3(x)$	1.17E+04（5.79E+03）	9.18E+04（5.71E+04）	7.47E+03（1.15E+03）	3.15E+08（4.01E+07）	**2.05E+03（8.28E+03）**
$f_4(x)$	5.13E+00（6.20E−01）	9.14E−05（6.01E−05）	4.01E−05（7.23E−07）	1.01E+00（9.10E+00）	**1.28E−07（7.02E−08）**
$f_5(x)$	3.07E+03（2.06E+03）	6.75E+02（2.19E+02）	7.78E+02（5.14E+02）	1.08E+03（4.05E+03）	**1.71E+02（2.69E+02）**
$f_6(x)$	1.67E+02（2.13E+00）	1.44E+02（4.27E+01）	7.28E+00（9.91E+00）	3.28E+07（1.85E+05）	**2.72E−01（8.14E−01）**
$f_7(x)$	9.18E−03（3.01E−03）	8.06E−03（3.67E−03）	4.59E−03（3.18E−03）	8,24E+00（1.33E−01）	**4.06E−03（2.59E−03）**
$f_8(x)$	2.09E+01（1.37E−02）	2.09E+01（2.18E−02）	2.01E+01（3.89E−02）	2.09E+01（4.17E−02）	**2.01E+01（1.88E−02）**
$f_9(x)$	5.37E−23（4.43E−24）	3.71E−19（2.01E−21）	3.25E−27（6.47E−29）	4.01E−15（6.22E−16）	**2.04E−33（5.01E−37）**
$f_{10}(x)$	7.23E+01（3.08E+01）	6.17E+01（5.68E+00）	4.83E+01（7.65E+00）	4.26E+01（5.18E+00）	**3.97E+01（2.61E+00）**
$f_{11}(x)$	5.77E+01（3.18E+01）	3.05E+01（2.17E+00）	2.99E+01（2.08E+00）	8.64E+01（7.21E+00）	**1.45E+01（5.84E+00）**
$f_{12}(x)$	1.05E+05（7.71E+04）	7.89E+02（8.59E+01）	6.18E+04（4.39E+03）	**7.61E+02（5.35E+01）**	2.37E+03（1.05E+03）
$f_{13}(x)$	3.44E+00（3.84E−01）	2.81E+00（5.64E−02）	3.14E+00（8.78E−02）	4.36E+01（2.58E+00）	**2.13E+00（3.82E−02）**
$f_{14}(x)$	4.47E+01（1.23E−01）	2.29E+01（4.38E−01）	2.09E+01（3.91E−01）	3.58E+02（7.69E+01）	**1.54E+01（2.03E−01）**

表 2　5 种算法在测试函数 f_1-f_{14} 上的 EV 平均值和标准差比较（$D=100$）

函数 \ 算法	jDE	SaDE	JADE	DEahcSPX	dn-DADE
	平均值（标准差）	平均值（标准差）	平均值（标准差）	平均值（标准差）	平均值（标准差）
$f_1(x)$	3.62E−16（4.91E−18）	4.98E−18（7.46E−19）	5.16E−21（3.88E−23）	5.94E−10（7.78E−12）	**6.81E−33（4.76E−37）**
$f_2(x)$	7.69E−06（5.46E−07）	3.16E−02（4.49E−03）	7.91E−07（5.04E−08）	2.88E+00（1.87E−01）	**2.73E−10（5.47E−12）**
$f_3(x)$	1.14E+07（5.83E+05）	6.49E+06（3.06E+03）	1.93E+06（5.58E+04）	2.58E+07（3.44E+06）	**8.43E+05（1.56E+04**
$f_4(x)$	4.78E+04（3.13E+03）	2.89E+04（5.98E+03）	3.07E+04（4.57E+03）	5.91E+04（1.31E+04）	**2.84E+03（1.16E+03）**
$f_5(x)$	1.08E+05（4.31E+03）	8.57E+04（5.08E+03）	5.36E+04（2.41E+03）	7.67E+04（3.58E+04）	**2.19E+04（3.18E+03）**
$f_6(x)$	4.92E+02（7.15E+00）	2.87E+02（9.06E+01）	8.96E+00（6.18E−01）	2.35E+03（4.48E+02）	**5.81E+00（4.56E−01）**
$f_7(x)$	1.05E−01（6.43E−02）	8.02E−02（4.14E−02）	7.84E−02（3.26E−02）	3.64E−01（2.28E−01）	**1.46E−02（7.81E−03）**
$f_8(x)$	2.21E+01（3.46E+00）	2.21E+01（5.18E+00）	**2.11E+01（8.97E−01）**	2.21E+01（7.64E+01）	2.11E+01（1.36E+00）
$f_9(x)$	9.98E+02（5.37E+02）	9.08E+02（4.26E+02）	8.07E+02（1.84E+01）	9.67E+02（5.04E+02）	**5.68E+02（3.75E+00）**
$f_{10}(x)$	5.76E+02（8.43E+01）	7.84E+02（2.61E+02）	3.55E+02（6.07E+01）	8.59E+02（4.65E+02）	**1.84E+02（4.43E+01）**
$f_{11}(x)$	5.83E+02（2.75E+01）	7.69E+02（2.24E+01）	1.58E+02（9.07E+00）	1.24E+03（6.28E+01）	**1.04E+02（7.81E+00）**
$f_{12}(x)$	6.05E+04（4.86E+03）	**8.81E+03（1.24E+03）**	4.59E+04（1.94E+04）	8.16E+04（3.83E+04）	3.47E+04（6.63E+03）
$f_{13}(x)$	4.72E+01（3.06E+01）	6.58E+00（3.59E+00）	**5.46E+00（3.95E+00）**	6.89E+01（5.90E+01）	2.17E+01（5.38E+00）
$f_{14}(x)$	3.32E+01（4.77E−01）	2.84E+01（2.97E−01）	2.26E+01（3.41E−01）	3.47E+01（5.54E−01）	**1.84E+01（1.59E−01）**

表 3　对测试函数 f_1-f_{14} 的 Wilcoxon 秩和检验（显著度水平 5%）的 P 值（$D=30$）

函数 \ 算法	jDE	SaDE	JADE	DEahcSPX	dn-DADE
$f_1(x)$	4.19E−17	5.82E−17	2.74E−22	7.96E−17	**NA**
$f_2(x)$	2.13E−11	5.26E−16	7.63E−23	1.69E−07	**NA**
$f_3(x)$	6.18E−04	2.84E−06	1.04E−04	3.53E−02	**NA**
$f_4(x)$	5.17E−04	6.62E−03	3.47E−04	9.18E−04	**NA**
$f_5(x)$	1.82E−02	5.47E−05	2.88E−04	8.19E−03	**NA**
$f_6(x)$	5.19E−10	4.98E−09	8.27E−10	1.82E−04	**NA**
$f_7(x)$	6.52E−13	7.71E−05	5.38E−04	4.19E−02	**NA**
$f_8(x)$	4.75E−12	6.82E−11	1.09E−07	7.57E−03	**NA**
$f_9(x)$	3.19E−11	5.48E−17	8.42E−22	2.14E−20	**NA**
$f_{10}(x)$	5.13E−15	7.22E−16	1.15E−17	2.18E−15	NA
$f_{11}(x)$	4.41E−07	6.92E−18	5.59E−18	2.97E−05	NA
$f_{12}(x)$	5.84E−18	3.72E−19	1.89E−04	**NA**	8.17E−07
$f_{13}(x)$	8.28E−16	2.03E−06	4.87E−16	5.22E−05	**NA**
$f_{14}(x)$	2.76E−06	4.39E−17	6.91E−18	3.29E−04	**NA**

表 4　对测试函数 f_1-f_{14} 的 Wilcoxon 秩和检验（显著度水平 5%）的 P 值（$D=100$）

函数 \ 算法	jDE	SaDE	JADE	DEahcSPX	dn-DADE
$f_1(x)$	2.97E−16	4.15E−17	6.88E−19	5.72E−10	**NA**
$f_2(x)$	5.41E−13	6.95E−20	1.37E−20	4.84E−12	**NA**
$f_3(x)$	5.89E−09	4.37E−08	2.17E−05	3.64E−09	**NA**
$f_4(x)$	1.07E−07	9.18E−05	9.78E−07	6.47E−05	**NA**
$f_5(x)$	2.68E−04	5.39E−06	5.02E−07	6.85E−07	**NA**
$f_6(x)$	5.14E−10	4.28E−07	8.96E−14	2.28E−08	**NA**
$f_7(x)$	8.42E−13	3.14E−07	6.11E−05	7.87E−11	**NA**
$f_8(x)$	1.58E−07	5.76E−11	**NA**	3.46E−07	6.73E−12
$f_9(x)$	7.82E−11	5.68E−17	1.43E−19	2.83E−14	**NA**
$f_{10}(x)$	2.49E−15	5.96E−11	3.17E−17	6.71E−12	NA
$f_{11}(x)$	5.77E−06	4.03E−20	1.89E−13	8.74E−05	NA
$f_{12}(x)$	4.26E−17	**NA**	2.68E−14	1.56E−10	5.13E−13
$f_{13}(x)$	8.82E−20	1.09E−10	**NA**	5.29E−05	7.36E−15
$f_{14}(x)$	4.91E−09	2.78E−13	3.19E−17	6.59E−11	**NA**

由表1、表2可见，对于所有测试函数，无论是30维还是100维，*dn*-DADE算法在大多数函数上的优化结果都明显优于其他比较算法。从表1、表3可以看出，对于30维的问题，除了在函数 f_{12} 的求解上DEahcSPX算法和SaDE算法的优化结果好于 *dn*-DADE 算法外，其他13个函数上 *dn*-DADE 算法均表现最优，尤其在函数 f_1 和 f_9 上表现较为突出。其中JADE算法在函数 f_7 和 f_8 上的求解结果与 *dn*-DADE 算法相当，但 *dn*-DADE 算法的标准差都小于JADE算法，表明其具有更好的稳定性。表2、表4的实验结果显示，*dn*-DADE 算法在函数100维时仍保持了良好的优化性能，拥有较高的求解精度，该算法在函数 f_8、f_{12} 和 f_{13} 上表现次优，其中SaDE算法对函数 f_{12} 的优化性能最高，在函数 f_8 和 f_{13} 的求解上JADE算法则表现最好，而其在 f_1-f_7、f_9-f_{11} 和 f_{14} 这11个函数上继续取得了最好的求解结果。由此可见，*dn*-DADE 算法能够有效地处理旋转、含噪问题及其混合函数，其在绝大部分函数上取得的最优解质量均优于其他比较算法，性能表现也没有因为函数维数的升高而降低。

图2—图5是各算法对30维函数运行50次后得到的收敛过程曲线。由于篇幅所限，只给出 f_4、f_6、f_7 和 f_9 这4个函数的收敛比较图，图中的纵坐标采用EV值以10为底数的对数表示。从图中可以看出，对于单峰函数 f_4，*dn*-DADE 算法的收敛速度相对其他比较算法拥有明显的优势；而对于较为复杂的多峰函数 f_6 和 f_7，*dn*-DADE 算法与JADE算法的收敛速度均快于另3种算法，两者都采用“贪婪性”的变异策略，在进化前期具有相似的收敛特性，随着进化代数的增加，*dn*-DADE 算法的收敛开始明显加快，特别地，对于函数 f_9，*dn*-DADE 算法在运行初始阶段的收敛速度较慢，但参数自适应的调整不断引导种群朝着全局最优解的进化方向收敛，同时新的变异策略更加快了其收敛，因此最后 *dn*-DADE 算法的收敛速度超过了其他算法，并且取得了更好的收敛精度（见表1）。

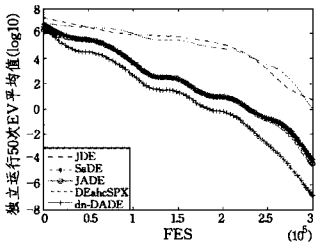


图2　5种算法在 f_4（D=30）上的收敛曲线图

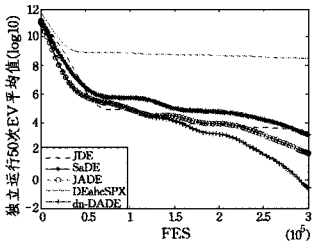


图3　5种算法在 f_6（D=30）上的收敛曲线图

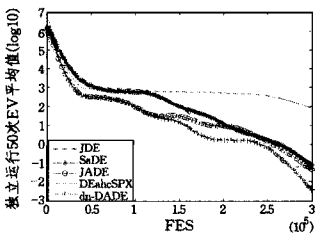


图4　5种算法在 f_7（D=30）上的收敛曲线图

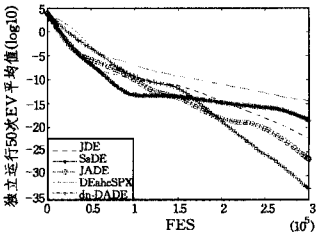


图5　5种算法在 f_9（D=30）上的收敛曲线图

4.4　*dn*-DADE 算法改进性能分析

本文提出的 *dn*-DADE 算法从变异策略和参数设置两方面对标准DE算法进行了改进，分别针对变异策略和参数提出了动态自适应方案。为了更好地展示改进部分的有效性，选用以下算法作为比较：1）*dn*-DE 算法：仅采用 *dn*-DADE 算法的变异策略，无参数自适应控制（设置 $F_{dn}=0.7$，$CR_{dn}=0.5$，$\sigma^2_{dn}=0.01$ 不变）；2）DADE 算法：仅参数进行自适应调整，变异策略采用一般性的 DE/rand/1；3）*dnc*-DADE 算法：*dn* 保持不变（$dn=NP/4$），其他设置与 *dn*-DADE 算法相同。基于所有测试函数（D=30）的4种算法均进行50次独立运行，其平均最优值（最小误差值）和标准差如表5所列。

表5　4种算法在测试函数 f_1-f_{14} 上的EV平均值和标准差比较（D=30）

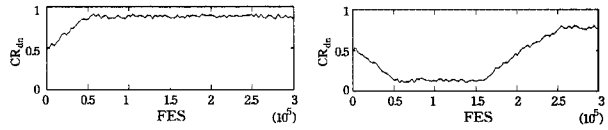
函数 \ 算法	dn-DE	DADE	dnc-DADE	dn-DADE
	平均值（标准差）	平均值（标准差）	平均值（标准差）	平均值（标准差）
$f_1(x)$	2.19E−34 (4.83E−39)	5.07E−37 (7.83E−35)	3.58E−43 (6.78E−46)	**7.25E−58 (1.83E−60)**
$f_2(x)$	4.71E−12 (6.24E−13)	8.05E−12 (2.76E−12)	1.49E−15 (6.67E−17)	**5.17E−26 (4.64E−26)**
$f_3(x)$	4.47E+04 (2.19E+04)	3.68E+04 (1.74E+05)	6.92E+03 (8.74E+03)	**2.05E+03 (8.28E+03)**
$f_4(x)$	9.74E−03 (5.13E−05)	8.16E−04 (1.98E−03)	5.62E−05 (4.11E−06)	**1.28E−07 (7.02E−08)**
$f_5(x)$	4.95E+03 (5.12E+02)	3.28E+03 (2.91E+03)	3.15E+03 (7.64E+02)	**1.71E+02 (2.69E+02)**
$f_6(x)$	8.67E+02 (2.81E+01)	4.39E+01 (7.64E+00)	6.17E+00 (2.36E+00)	**2.72E−01 (8.14E−01)**
$f_7(x)$	5.87E−02 (4.96E−02)	4.91E−02 (4.83E−02)	7.16E−03 (3.89E−03)	**4.06E−03 (2.59E−03)**
$f_8(x)$	2.28E+01 (4.51E−01)	2.34E+01 (3.89E−01)	2.11E+01 (7.89E−02)	**2.01E+01 (1.88E−02)**
$f_9(x)$	8.27E−05 (9.24E−07)	5.51E−08 (3.47E−08)	7.49E−10 (7.16E−11)	**2.04E−12 (5.01E−14)**
$f_{10}(x)$	1.86E+02 (5.17E+01)	7.36E+01 (2.78E+01)	6.64E+01 (1.96E+01)	**3.97E+01 (2.61E+00)**
$f_{11}(x)$	1.59E+01 (1.28E+01)	2.01E+01 (9.57E+00)	1.57E+01 (1.06E+01)	**1.45E+01 (5.84E+00)**
$f_{12}(x)$	3.57E+05 (2.18E+04)	6.28E+04 (3.88E+03)	1.64E+04 (5.91E+03)	**2.37E+03 (1.05E+03)**
$f_{13}(x)$	5.91E+03 (2.87E+03)	3.24E+03 (7.18E+02)	2.47E+01 (1.82E+00)	**2.13E+00 (3.82E−02)**
$f_{14}(x)$	9.61E+01 (7.28E+00)	9.79E+01 (3.54E+01)	9.38E+01 (3.96E+00)	**1.54E+01 (2.03E−01)**

表5的测试结果显示，相比只改进一部分的变种算法，*dn*-DADE 算法具有明显的性能优势，进一步地说明了 DE/current-to-*dn*best/1 变异策略和参数自适应策略的缺一不可和相互作用：新的变异策略在算法运行过程中能够动态地平衡算法的探索和开采能力，并通过精英解的选择对寻优方向加以引导，避免算法随机选择导致的搜索盲目性。虽然在后期由于 *dn* 的变化更倾向于“贪婪性”策略，可能导致算法陷入局部最优，但通过参数自适应调整可以有效地提高种群多样性，帮助算法寻找到合适的精英解，从而加快种群向有效进化方向的收敛，提高算法的收敛速度和稳定性。

4.5　参数分析

如3.2节所述，DE算法的性能相对于控制参数 F 和 CR 的设置是很敏感的，对不同的问题需要进行不同的设置。为达到更好的优化效果和减少人为设置的困难，本文通过对 F_{dn} 和 CR_{dn} 的动态调整设计了新的自适应算子。图6显示了 CR_{dn} 在优化函数 f_6 和 f_9 过程中的变化轨迹。交叉概率因子 CR 的取值与问题的特性有关，因此选取两个不同性质的30

维函数来观察 CR_{dn} 的自适应情况，其中 f_6 代表自变量相关(non-separable)问题，f_9 代表自变量独立(separable)问题。从图中可以看出，针对不同问题，CR_{dn} 在搜索的不同阶段呈现不同的变化趋势，有效地对参数起到了自适应调整的作用。



(a) f_6 (b) f_9

图 6 CR_{dn} 的变化趋势图

另外，由 3.2.2 节可知，为了使实际产生的 $F_{i,G}$ 值尽可能落在其限定范围内以减少截断次数，根据柯西分布的特点对 F_{dn} 的取值范围设置了调整量 $\theta r(\theta=1,2,3,\cdots,r=0.05)$。$\theta$ 取值对 $F_{i,G}$ 值超限概率的影响如表 6 所列。显而易见，随着 θ 值的增大，F_{dn} 的取值范围越来越小，同时 $F_{i,G}$ 值的超限概率越来越低。由于 $F_{i,G}$ 值的超限概率与最大进化代数 G_{max} 无关，仅与 F_{dn} 的初始值和 θ 的取值相关，综合考虑，在设定 F_{dn} 初始值 0.7 的条件下，本文选取 $\theta=2$，以平衡 F_{dn} 的取值范围和超限概率的关系。

表 6 θ 的取值与 $F_{i,G}$ 值超限概率的关系

θ 值	超限概率
$\theta=1$	25%
$\theta=2$	15%
$\theta=3$	10%
$\theta=4$	8%

结束语 DE 算法在处理复杂函数最优化时存在后期收敛精度不高、速度较慢以及对参数设置敏感的问题，为此本文提出了一种新的变异策略 DE/current-to-*dn*best/1 和新的参数自适应策略，对标准 DE 算法进行了综合改进，构成一种新型的基于动态自适应策略的 *dn*-DADE 算法。为验证 *dn*-DADE 算法的整体先进性和高效性，选取多种现有先进 DE 改进算法在 14 个 Benchmark 测试函数上进行对比实验，结果显示该算法具有更好的求解精度、收敛速度和稳定性。同时，通过将本文提出的 *dn*-DADE 算法与其本身的变种算法进行比较，进一步说明了改进的有效性。

参 考 文 献

[1] 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

[2] Price K, Storn R. Differential Evolution-A practical approach to global optimization [M]. Berlin, Germany: Springer Verlag, 2006: 133-152

[3] Das S, Abraham A, Konar A. Automatic clustering using an improved differential evolution algorithm [J]. IEEE Transaction on Systems, Man and Cybernetics, 2008, 38(1): 218-236

[4] 韩敏，王明慧，范剑超. 基于改进差分进化算法的在线轨迹优化[J]. 控制与决策，2012，27(2)：247-251

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

[6] 袁俊刚，孙治国，曲广吉. 差异演化算法的数值模拟研究[J]. 系统仿真学报，2007，19(20)：4646-4648

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

[8] Brest J, Greiner S, Boskovie B. 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

[9] Mallipeddi R, Suganthana P, Pan Q. Differential evolution algorithm with ensemble of parameters and mutation strategies [J]. IEEE Transactions on Evolutionary Computation, 2011, 11: 1679-1696

[10] Salman A, Engelbrecht A P, Omran M G H. Empirical analysis of self-adaptive differential evolution [J]. European Journal of Operational Research, 2007, 183(2): 785-804

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

[12] Mezura-Montes E, Velázquez-Reyes J, Coello C A C. A comparative study of differential evolution variants for global optimization [C] // Proceedings of Genetic Evolutionary Computation Conference (GECCO-2006). 2006: 485-492

[13] Suganthan P N, Hansen N, Liang J J, et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization[R]. Nanyang Technological University, Singapore, 2005

[14] Noman N, Iba H. Accelerating differential evolution using an adaptive local search [J]. IEEE Transaction on Evolutionary Computation, 2008, 12(1): 107-125

(上接第 235 页)

[18] Pang B, Lee L, Vaithyanathan S. Thumbs Up? Sentiment Classification Using Machine Learning Techniques[C] // Proceedings of EMNLP-2002. 2002: 79-86

[19] Pang Bo, Lee L. Opinion mining and sentiment analysis [J]. Foundations and Trends in Information Retrieval, 2008, 2(1/2): 1-135

[20] Tanaka Y, Takamura H, Okumura M. Extraction and classification of facemarks[C] // Proceedings of the 10th international conference on Intelligent user interfaces (IUI). New York, NY, USA: ACM, 2005: 28-34

[21] Aoki S, Uchida O. A method for automatically generating the emotional vectors of emoticons using weblog articles[C] // Proceedings of the 10th WSEAS international conference on Applied computer and applied computational science (ACACOS). Venice, Italy: WSEAS Press, 2011: 132-136

[22] Paltoglou G, Thelwall M. Twitter, MySpace, Digg: Unsupervised Sentiment Analysis in Social Media[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2012, 3(4): 1-19