一种求解约束优化问题的自适应差分进化算法

阎大海[1],李元香[1],龚文引[2],何国良[1]
(1. 武汉大学软件工程国家重点实验室,武汉大学计算机学院,湖北武汉 430072;
2. 中国地质大学(武汉)计算机学院,湖北武汉 430074)

摘 要: 自适应算子选择方式已被用于差分进化算法求解全局优化问题及多目标优化问题,然而在求解约束优化时难于为自适应算子选择方式找到一种方式来恰当分配信用. 为此,本文提出了一种基于混合种群的自适应适应值方式来对约束优化问题中变异策略进行信用分配并采用概率匹配方法自适应选择差分变异策略,同时对算法变异缩放因子与交叉率进行自适应设置提高算法的成功率. 实验结果表明算法在求解约束优化问题相比于 CODEA/OED, ATMES,εBBO-dm,COMDE 以及 εDE 算法有较高的收敛精度及收敛速度,同时验证了自适应方式的有效性. 该算法可用于预报、质量控制、会计过程等科学和工程应用领域.

关键词: 约束优化; 差分进化算法; 自适应; 信用分配; 概率匹配
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2016)10-2535-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.10.036

An Adaptive Differential Evolution Algorithm for Constrained Optimization Problems

XIA Da-hai[1], LI Yuan-xiang[1], GONG Wen-yin[2], HE Guo-liang[1]
(1. *College of Computer Science, Wuhan University. Wuhan, Hubei* 430072, *China*;
2. *School of Computer Science, China University of Geosciences. Wuhan, Hubei* 430074, *China*)

Abstract: The adaptive operator selection method is used to solve the global optimization problem and multi-objective optimization problem of differential evolution algorithm. However, it is difficult to find a way to properly allocate credit for the adaptive operator selection in solving the constrained optimization problem. In order to realize the adaptive strategy selection in differential evolution, we present a combined population based adaptive fitness method to achieve the credit assignment of mutate strategies for constrained optimization problems and use probability matching method to select the mutate strategy adaptively. And we also set the mutation scaling factor and the crossover rate adaptively to improve the success rate of the algorithm. Experimental results show that the algorithm has higher accuracy and convergence speed comparing to CODEA/OED, ATMES, εBBO-dm, COMDE and εDE. We also test and verify the effectiveness of the adaptive method. The algorithm can be used in forecasting, quality control, accounting process, and other scientific and engineering applications.

Key words: constrained optimization; differential evolution algorithm; adaptation; credit assignment; probability matching

1 引言

差分进化算法需要根据实际问题需要设置相关参数(种群规模 NP,变异缩放因子 F,交叉率 CR)以及选择合适变异策略,这通常需要进行反复试验才能得到最佳设置. 而且随着算法搜寻的解空间区域不同,不同区域所需要的参数及变异策略也不相同[1]. 因此自适应选择变异策略及设置参数是一种较好的方式.

自适应算子选择(Adaptive Operator Selection, AOS)开始被用于遗传算法[2,3],近来被引入差分进化算法解决全局优化[4,5]以及多目标优化[6]问题,就作者所知该

收稿日期:2015-11-20; 修回日期:2016-01-07;责任编辑:覃怀银
基金项目:国家重大仪器专项(No. 2011YQ170065.4);国家自然科学基金(No. 61573324)

类方法尚未用于解决约束优化问题(Constrained Optimization Problems,COPs). 文献[5]提出了一种利用概率匹配(probability matching)方式自适应选择算子的差分进化算法,受此启发本文提出了一种用于解决约束优化的自适应算子选择方法. 自适应算子选择需要解决两个问题:一是如何根据操作算子在近期搜寻过程中的表现给出相应信用分配(credit assignment),本文为此提出了一种基于混合种群的自适应适应值的方法来分配信用;二是如何根据分配的信用值来选择操作算子,本文使用了概率匹配方式来选择. 此外,对于算法的交叉率 CR 和缩放因子 F,本文采用了 JADE 算法[7]算法机制来对参数做自适应设置. 新的自适应差分约束算法称为 PJAD-CDE 算法,通过标准测试函数测试测试表明算法在求解约束优化问题时有较好的收敛精度及收敛速度并验证了自适应方法的有效性.

2 约束优化问题定义

不失一般性,约束优化问题可按照以下规则定义:

$$\begin{aligned} &\text{minimize } f(x), && x\in D\\ &\text{subject to } g_j(x)\leqslant 0, && j=1,\cdots,q\\ &\qquad h_j(x)=0, && j=q+1,\cdots,m \end{aligned} \tag{1}$$

其中,$D=\prod_{i=1}^{n}[a_i,b_i]\subseteq R^n$,$x=(x_1,x_2,\cdots,x_n)$ 且 $x_i\in[a_i,b_i]$,$f:D\to R$ 称为目标函数,$g_j(x)$ 是第 j 个非等式约束条件,$h_j(x)$ 是第$(j-q)$个等式约束条件. 同时,令 $S=\{X\mid X\in D\wedge g_i(x)\leqslant 0\wedge h_j(x)=0\}$ 表示可行域.

一般将等式约束转换为不等式约束处理

$$|h_j(x)|-\delta\leqslant 0,\ j\in\{q+1,\cdots,m\} \tag{2}$$

δ 为一个正的容忍值. 一个解 x 到第 j 个约束的距离定义为

$$G_j(x)=\begin{cases}\max\{0,g_j(x)\}, & 1\leqslant j\leqslant q\\ \max\{0,|h_j(x)-\delta|\}, & q+1\leqslant j\leqslant m\end{cases} \tag{3}$$

解 x 到可行区域边界的距离定义为 $G(x)$,反映了 x 违反约束值的大小

$$G(x)=\sum_{j=1}^{m}G_j(x) \tag{4}$$

3 标准差分进化算法

标准差分进化算法利用个体混合变异杂交产生后代个体,当后代个体优于父代个体时替换父代个体. 差分进化算法中的主要参数包括种群规模 NP,变异缩放因子 F,以及交叉率 CR.

变异算子是差分算法的主要算子. 对于种群个体 x_i,经过变异算子变异后产生的相应变异个体记为 v_i,常用的差分变异算子包括:

(1)"DE/rand/1":

$$v_i=x_{r1}+F(x_{r2}-x_{r3}) \tag{5}$$

(2)"DE/rand/2":

$$v_i=x_{r1}+F(x_{r2}-x_{r3})+F(x_{r4}-x_{r5}) \tag{6}$$

(3)"DE/rand-to-best/2":

$$v_i=x_{r1}+F(x_{\text{best}}-x_{r1})+F(x_{r2}-x_{r3})+F(x_{r4}-x_{r5}) \tag{7}$$

(4)"DE/current-to-rand/1":

$$v_i=x_i+F(x_{r1}-x_i)+F(x_{r2}-x_{r3}) \tag{8}$$

其中 $x_{r1},x_{r2},x_{r3},x_{r4},x_{r5}$ 是从整个种群中随机选取的互不相同个体,x_{best} 为整个种群中适应值最好的个体.

4 PJAD-CDE 算法的变异策略自适应选择

本文和文献[5]一样选取式(5)~(8)的变异策略组成变异策略池来进行自适应选择. 自适应选择实现包括:(1)信用分配机制,即如何衡量不同的变异策略在搜寻中的优劣;(2)策略选择机制,即采用何种方式选择变异策略. 下面介绍两者的实现.

4.1 信用分配机制

信用分配机制包括:(1)如何衡量由单个策略造成的适应值变化;(2)如何根据适应值变化分配恰当的信用值. 对于(1),约束优化问题需要同时考虑个体的目标函数值和违约值,因此需要找到一个恰当的标准来衡量个体的适应值变化. 文献[8]提出了一种在约束优化中的个体排序标准,受此启发本文提出了一种混合种群的自适应适应值(combined population based adaptive fitness)方法来衡量个体的适应值变化. 将当前种群记为 P_t,下代种群记为 P_{t+1},将 P_t 和 P_{t+1} 混合之后的种群记为 Pc. 种群 Pc 分为三种状态:不可行状态,半可行状态,可行状态. 计算个体 x_i 的适应值 $\text{Fit}(x_i)$ 方法如下:

(1)不可行状态. 该状态下种群中只有不可行个体,因此找到可行个体是最重要的,所以个体适应值按照违约值 $G(x_i)$ 计算.

$$\text{Fit}(x_i)=G(x_i) \tag{9}$$

(2)半可行状态. 在半可行状态下,种群中同时包含可行个体和非可行个体,按照文献[9]中的自适应适应值转换(Adaptive Fitness Transformation,AFT)方法来计算个体适应值.

将种群 Pc 分为可行解集合(Z_1)和非可行解集合(Z_2),个体 x_i 的目标函数值 $f(x_i)$ 按以下公式转换

$$f'(x_i)=\begin{cases}f(x_i), & i\in Z_1\\ \max\{\varphi\cdot f(x_{\text{best}})+(1-\varphi)\cdot f(x_{\text{worst}}),f(x_i)\}, & i\in Z_2\end{cases} \tag{10}$$

φ 为种群 Pc 可行解所占比率,x_{best} 与 x_{worst} 为 Z_1 的最好

与最差个体.$f'(x)$标准化为

$$f_{\text{nor}}(x_i)=\frac{f'(x_i)-\min\limits_{j\in Z_1\cup Z_2}f'(x_j)}{\max\limits_{j\in Z_1\cup Z_2}f'(x_j)-\min\limits_{j\in Z_1\cup Z_2}f'(x_j)} \tag{11}$$

用式(4)来计算个体的违约值并将其标准化

$$G_{\text{nor}}(x_i)=\begin{cases}0, & i\in Z_1\\ \dfrac{G(x_i)-\min\limits_{j\in Z_2}G(x_j)}{\max\limits_{j\in Z_2}G(x_j)-\min\limits_{j\in Z_2}G(x_j)}, & i\in Z_2\end{cases} \tag{12}$$

个体适应值按以下公式计算

$$\text{Fit}(x_i)=f_{\text{nor}}(x_i)+G_{\text{nor}}(x_i) \tag{13}$$

(3) 可行状态. 该状态下种群中只有可行解,适应值由目标函数值$f(x_i)$决定.

$$\text{Fit}(x_i)=f(x_i) \tag{14}$$

种群P_t中的个体x_i适应值记为$\text{Fit}(x_i)$,对应试验个体x_{i+1}适应值记为$\text{Fit}(x_{i+1})$,适应值变化$\text{FI}_i=\text{Fit}(x_i)-\text{Fit}(x_{i+1})$. 由于在进化早期低劣个体较多适应值提高较大,在进化晚期则反之,因此直接使用适应值的差来计算适应值提高FI_i会使进化早期效果较好的策略在进化中占据优势,所以对$F(x_i)$进行标准化.

$$\text{Fit}_{\text{normal}}(x_i)=\frac{\text{Fit}(x_i)-\text{Fit}_{\text{best}}(x_i)}{\text{Fit}_{\text{worst}}(x_i)-\text{Fit}_{\text{best}}(x_i)} \tag{15}$$

个体适应值提高FI_i按以下方式计算

$$\text{FI}_i=\text{Fit}_{\text{normal}}(x_i)-\text{Fit}_{\text{normal}}(x_{i+1}) \tag{16}$$

将S_a记为策略$a(a=1,\cdots,k)$在第t代的适应值提高集合. 则策略a在第t代的信用分配$r_a(t)$按集合S_a的平均值计算:

$$r_a(t)=\frac{\sum\limits_{i=1}^{|S_a|}S_a(i)}{|S_a|} \tag{17}$$

4.2 策略选择机制

在策略选择机制上,本文采用概率匹配方法来选择策略.$r_a(t)$为策略a在第t代所获取的信用,$q_a(t)$为策略的已知质量,则该策略在下代质量$q_a(t+1)$的更新公式为:

$$q_a(t+1)=q_a(t)+\alpha[r_a(t)-q_a(t)] \tag{18}$$

$\alpha\in(0,1]$为适应率. 策略选择概率按以下公式更新

$$p_a(t+1)=p_{\min}+(1-k\cdot p_{\min})\frac{q_a(t+1)}{\sum\limits_{i=1}^{k}q_i(t+1)} \tag{19}$$

$p_{\min}\in(0,1)$为最小选择概率.

5 PJAD-CDE 算法中的参数自适应设置

PJAD-CDE 算法的变异缩放因子F与交叉率 CR 的自适应设置按照类似 JADE 算法机制进行. 记F_i^a为个体x_i在使用策略$a\in(1,2,\cdots,k)$时的变异因子,F_i^a由定位参数为μ_F^a规模参数为 0.1 的柯西分布生成:

$$F_i^a=\text{cauchy}(\mu_F^a,0.1) \tag{20}$$

当F_i^a大于 1 或小于等于 0 时重新生成. 策略a在第t代成功的变异因子集合记为S_F^a,μ_F^a在每一代按照以下公式更新:

$$\mu_F^a=(1-c)\cdot\mu_F^a+c\cdot\text{mean}_L(S_F^a) \tag{21}$$

$\text{mean}_L(\cdot)$为 Lehmer 平均值.

对于交叉率 CR,记CR_i^a为个体x_i在使用策略$a\in(1,2,\cdots,k)$时的交叉率,CR_i^a由期望为μ_{CR}^a方差为 0.1 的正态分布生成:

$$\text{CR}_i^a=\text{normal}(\mu_{\text{CR}}^a,0.1) \tag{22}$$

CR_i^a大于 1 或小于 0 时重新生成.μ_{CR}^a的生成公式与式(21)相同.

6 PJAD-CDE 算法流程

PJAD-CDE 算法具体流程如下,"DE/rand-to-best/2"策略中的 best 个体按如下方式选择:(1)可行个体优于不可行个体;(2)不可行个体中违反约束值$G(x)$小的个体占优;(3)可行个体中目标函数值$f(x)$小的个体占优. 在处理等式约束时,按文献[9]中类似机制将等式约束转换为不等式约束处理,t为进化代数

$$\delta_{t+1}=\begin{cases}\dfrac{\delta_t}{\delta'}, & \text{if } \delta_t>0.0001\\ 0.0001, & \text{otherwise}\end{cases} \tag{23}$$

δ的初始值经实验后设置为$n\cdot(\log_{10}(\max\limits_i(u_i-l_i))+4)$,$u_i$与$l_i$为$x$在维度$i$的上界与下界.$\delta'$设置为 1.015.

算法 1　PJAD-CDE 算法

输入:搜索空间R,目标函数f,违约函数G;
输出:最优解;

```
(1)  初始化种群,评估个体的目标函数值与违约值
(2)  对每个策略 a,设置 q_a(t)=0,p_a(t) = 1/k
(3)  while 不满足停止条件
(4)     for i=1 to NP do
(5)        按轮盘赌方式选择策略 SI_i
(6)        随机选择个体 r_1≠r_2≠r_3≠r_4≠r_5≠i,生成 j_rand = rndint(1,D)
(7)        利用式(20)、(22)生成 F_i^a 与 CR_i^a
(8)        for j=1 to D do
(9)           if rnreal_i[0,1) < CR_i^a or j = =j_rand then
(10)             使用策略 SI_i 生成个体 u_ij
(11)          end if
(12)       end for
(13)    end for
(14)    for i=1 to NP do
(15)       评估后代 u_i
(16)       if F(u_i)≤F(x_i) then
(17)          用式(16)计算 FI_i 并用个体 u_i 替代个体 x_i
(18)       else
```

(19) $FI_i = 0$
(20) end if
(21) $S_{Sli} = FI_i$
(22) end for
(23) 根据式(17)计算每个策略的报酬 $r_a(t)$
(24) 根据式(18)更新每个策略的质量 $q_a(t)$
(25) 根据式(19)更新每个策略的选择概率 $p_a(t)$
(26) 根据式(21)更新 μ_F^a 与 μ_{CR}^a
(27) $t = t + 1$
(28) end while
(29) 输出当前最优解

7 实验结果及分析

为了验证 PJAD-CDE 算法的有效性,本文选取了 13 个约束优化测试函数来测试,测试函数详见文献[10]. 将 PJAD-CDE 算法与 ATMES[9], CODEA/OED[11], εDE[12], COMDE[13], εBBO-dm[14] 5 种约束算法进行比较. PJAD-CDE 算法参数设置如下:$k=4$, $p_{\min}=0.05$, $\alpha=0.3$, $c=0.1$.

7.1 PJAD-CDE 与其它算法的性能比较

将 PJAD-CDE 算法与 CODEA/OED, ATMES, εBBO-dm, COMDE, εDE5 种算法进行比较, PJAD-CDE, CODEA/OED, εBBO-dm, ATMES 均独立运行 30 次,最大函数评价次数为 240000 次. εDE 运行 50 次,最大函数评价次数为 2000000 次. COMDE 运行 30 次,最大函数评价次数为 200000 次.

由表 1 看到就收敛精度而言,PJAD-CDE 在除函数 $g02$ 外的其余 12 个测试函数中的运行得到的最优值,最差值以及平均值均小于或等于其它 5 种算法的运行结果. 函数 $g02$ 为高维多峰函数,主要考察算法的寻优能力. 在该函数的运行结果上,PJAD-CDE 运行结果略差于 εDE,好于其它几种算法,但考虑到 εDE 的函数评价次数远大于 PJAD-CDE,可以认为 PJAD-CDE 与 εDE 在 $g02$ 上的表现至少是相当的,表明 PJAD-CDE 的搜索能力强于或者不弱于其它几种算法. 在含有等式约束的函数 $g03$, $g05$, $g11$, $g13$ 的运行结果中,函数 $g03$ 和 $g05$ 只有 PJAD-CDE 找到了最优解. 对于函数 $g11$ 和 $g13$PJAD-CDE 与 εBBO-dm, COMDE 均在每次运行中找到了最优解. 显然 PJAD-CDE 相对于其它 5 种算法有更好的解决含有等式约束的问题的能力. 对于函数 $g01$, $g04$, $g06-g10$ 以及 $g12$, PJAD-CDE 与 εBBO-dm 每次均能找到最优解,优于其它 3 种算法. 在鲁棒性上,PJAD-CDE 在函数 $g01$, $g03$, $g05-g08$, $g11-g13$ 中的方差均小于等于其它 5 种算法,显然 PJAD-CDE 在鲁棒性上占优. 通过以上对比可以看出在寻优精度及稳定性上 PJAD-CDE 优于其它算法.

表 1 算法运行结果比较

函数/最优解	取值状态	PJAD-CDE	CODEA/OED	ATMES	εBBO-dm	εDE	COMDE
$g01/-15.000$	最好值	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**
	平均值	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**
	最差值	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**
	方差	**0**	**0**	1.6E-14	7.8E-20	**0**	1.97E-13
$g02/-0.803619$	最好值	**-0.803619**	**-0.8036019**	-0.803388	**-0.803619**	**-0.803619**	**-0.803619**
	平均值	-0.802587	-0.790908	-0.790148	-0.796513	**-0.803004**	-0.801238
	最差值	**-0.792608**	-0.761532	-0.756986	-0.792570	**-0.792608**	-0.785265
	方差	3.11E-03	7.90E-03	1.3E-02	2.29E-05	**2.47E-03**	5.0E-03
$g03/-1.0005$	最好值	**-1.0005**	-1.000	-1.000	-1.000	-1.000	-1.000000049
	平均值	**-1.0005**	-1.000	-1.000	-1.000	-1.000	-1.000000027
	最差值	**-1.0005**	-1.000	-1.000	-1.000	-1.000	-0.99999994
	方差	**0**	5.80E-09	5.9E-05	0	3.9E-06	3.026E-08
$g04/-30665.539$	最好值	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	平均值	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	最差值	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	方差	1.7E-12	1.20E-11	7.4E-12	**0**	2.1E-05	**0**

续表

函数/最优解	取值状态	PJAD-CDE	CODEA/OED	ATMES	εBBO-dm	εDE	COMDE
*g*05/5126.4967	最好值	**5126.4967**	5126.498	5126.498	5126.4981	5126.498	5126.498109
	平均值	**5126.4967**	5126.498	5127.648	5126.4981	5126.498	5126.498109
	最差值	**5126.4967**	5126.498	5135.256	5126.4981	5126.498	5126.498109
	方差	**0**	7.20E-12	1.8E+00	**0**	1.7E-05	**0**
*g*06/-6961.8138	最好值	**-6961.8138**	**-6961.814**	**-6961.814**	**-6961.8138**	**-6961.814**	**-6961.813875**
	平均值	**-6961.8138**	**-6961.814**	**-6961.814**	**-6961.8138**	**-6961.814**	**-6961.813875**
	最差值	**-6961.8138**	**-6961.814**	**-6961.814**	**-6961.8138**	**-6961.814**	**-6961.813875**
	方差	**0**	3.90E-11	4.6E-12	**0**	2.3E-08	**0**
*g*07/24.3062	最好值	**24.3062**	**24.306**	**24.306**	**24.3062**	**24.306**	**24.306209**
	平均值	**24.3062**	**24.306**	24.316	24.3062	**24.306**	**24.306209**
	最差值	**24.3062**	**24.306**	24.359	24.3062	24.306	24.306209
	方差	**6.3E-15**	1.10E-05	1.1E-02	4.5E-11	6.3E-06	4.7E-07
*g*08/-0.095825	最好值	**-0.095825**	**-0.095825**	**-0.095825**	**-0.096825**	**-0.095825**	**-0.095825**
	平均值	**-0.095825**	**-0.095825**	**-0.095825**	**-0.096825**	**-0.095825**	**-0.095825**
	最差值	**-0.095825**	**-0.095825**	**-0.095825**	**-0.096825**	**-0.095825**	**-0.095825**
	方差	**0**	1.60E-17	2.8E-17	2.19E-19	8.4E-17	9.00E-18
*g*09/680.63	最好值	**680.63**	**680.630**	**680.630**	**680.630**	**680.63**	**680.630057**
	平均值	**680.63**	**680.630**	680.639	**680.630**	**680.63**	**680.630057**
	最差值	**680.63**	**680.630**	680.673	**680.630**	**680.63**	**680.630057**
	方差	2.23E-14	3.50E-12	1.0E-02	**0**	2.2E-07	4.071E-13
*g*10/7049.248	最好值	**7049.248**	7049.521	7052.253	**7049.248**	**7049.248**	**7049.248020**
	平均值	**7049.248**	7072.167	7250.437	**7049.248**	**7049.248**	**7049.248077**
	最差值	**7049.248**	7155.754	7240.225	**7049.248**	**7049.248**	7049.248615
	方差	2.46E-13	3.7E+01	1.2E+02	**5.41E-18**	9.0E-06	1.5E-04
*g*11/0.7499	最好值	**0.7499**	0.75	0.75	**0.7499**	0.75	0.749999999
	平均值	**0.7499**	0.75	0.75	**0.7499**	0.75	0.749999999
	最差值	**0.7499**	0.75	0.75	**0.7499**	0.75	0.749999999
	方差	**0**	2.8E-07	3.4E-04	**0**	6.9E-14	**0**
*g*12/-1.000	最好值	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**
	平均值	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**	**-1.000**
	最差值	**-1.000**	**-1.000**	-0.994	**-1.000**	**-1.000**	**-1.000**
	方差	**0**	**0**	1.0E-03	**0**	**0**	**0**
*g*13/0.0539415	最好值	**0.0539415**	**0.0539415**	0.053950	**0.0539415**	0.053949	**0.0539415**
	平均值	**0.0539415**	0.1437424	0.053952	**0.0539415**	0.069631	**0.0539415**
	最差值	**0.0539415**	0.4397026	0.053959	**0.0539415**	0.438846	**0.0539415**
	方差	**0**	1.60E-01	1.3E-05	3.24E-17	7.6E-02	1.4E-17

为了对比算法的收敛速度,表 2 给出了 PJAD-CDE 算法与 CODEA/OED, ATMES, εBBO-dm 以及 COMDE 在函数评价次数为 240000 次到达最优解的成功率以及在最大函数评价次数为 500000 次时到达最优解所用的平均函数评价次数的比较,NA 表示达到最大评价次数仍未找到最优解,SR 表示成功率,NFEs 表示评估次数.

对于 εDE,其函数评价次数为 2000000 次远高于其它算法而寻优精度劣于 PJAD-CDE,显然 PJAD-CDE 时间复杂度小于 εDE. 由表 3 可以看到在成功率上 PJAD-CDE 算法优于其它 4 种算法. 在函数评价次数上,PJAD-CDE 在函数 $g01$,$g02$,$g07$,$g09$,$g10$,$g12$ 上优于其它算法. 为了进一步衡量算法在评价次数上的优劣,对 5 种算法给出 Friedman 排名. 由表 3 可得 PJAD-CDE 和 εBBO-dm 表现最好. 但在平均评价次数上,PJAD-CDE 明显优于 εBBO-dm. 总体来看,PJAD-CDE 在收敛速度上优于其它算法.

表 2 算法在成功率,平均评价次数的对比

函数	PJAD-CDE		CODEA/OED		ATMES		εBBO-dm		COMDE	
	NFEs	SR%	NFEs	SR%	NFEs	SR%	NFEs	SR%	NFEs	SR%
$g01$	32136	100	95320	100	89037	100	41680	100	54477	100
$g02$	50752	90	276480	36.67	377700	0	317360	23.33	76236	70
$g03$	85760	100	89420	100	95440	100	85040	100	75000	100
$g04$	27384	100	52770	100	47030	100	25680	100	25500	100
$g05$	84400	100	36320	100	24600	6.67	21040	100	100100	100
$g06$	15364	100	36570	100	45210	100	13920	100	7699	100
$g07$	54696	100	196420	100	215700	3.33	86160	100	118136	100
$g08$	1816	100	6440	100	3590	100	1920	100	1035	100
$g09$	26828	100	64030	100	67800	3.33	41040	100	32600	100
$g10$	89748	100	299600	0	*NA*	0	110480	100	141596	100
$g11$	76400	100	6390	100	4440	100	2320	100	16680	100
$g12$	3460	100	4130	100	9110	100	4320	100	3564	100
$g13$	93924	100	372160	16.67	21900	13.33	89680	100	104508	100
平均值	49436	99.23	118157	81.02	77042 + *NA*	55.89	64664	94.10	58240	97.69

表 3 算法在函数评价次数上的 Friedman 排名

算法	PJAD-CDE	CODEA/OED	ATMES	εBBO-dm	COMDE
排名值	2.307	4	4.15	2.307	2.462

由以上分析可知 PJAD-CDE 算法相对于 CODEA/OED,ATMES,εBBO-dm,εDE 以及 COMDE5 种算法在求解约束问题的收敛精度以及收敛速度上均有明显优势.

7.2 自适应选择策略的有效性验证

为了验证基于自适应选择策略的有效性,将 PJAD-CDE 算法的策略自适应部分改为运行策略“DE/rand/1”,“DE/rand/2”,“DE/rand-to-best/2”,“DE/current-to-rand/1”,算法分别定义为 PJAD-CDE1,PJAD-CDE2,PJAD-CDE3,PJAD-CDE4. 运行 30 次,函数最大评估次数为 500000 次. 运行结果如表 4,NFEs 表示评价次数,SR 表示成功率.

表 4 PJAD-CDE 与 PJAD-CDE1,PJAD-CDE2,PJAD-CDE3,PJAD-CDE4 在平均评价次数及成功率的对比

函数	PJAD-CDE		PJAD-CDE1		PJAD-CDE2		PJAD-CDE3		PJAD-CDE4	
	NFEs	SR%	NFEs	SR%	NFEs	SR%	NFEs	SR%	NFEs	SR%
$g01$	32136	100	36728	100	47220	100	23716	100	47992	100
$g02$	50752	90	60115	90	60595	60	23340	60	57712	20
$g03$	85760	100	86864	100	87812	100	86700	100	86700	100
$g04$	27384	100	31892	100	40148	100	21128	100	40056	100
$g05$	84400	100	84416	100	84496	100	84400	100	84492	100
$g06$	15364	100	16032	100	19516	100	13388	100	19308	100
$g07$	54696	100	88368	100	97756	100	41740	100	108348	100
$g08$	1816	100	2276	100	2884	100	1776	100	2796	100

续表

函数	PJAD-CDE		PJAD-CDE1		PJAD-CDE2		PJAD-CDE3		PJAD-CDE4	
	NFEs	SR%	NFEs	SR%	NFEs	SR%	NFEs	SR%	NFEs	SR%
*g*09	26828	100	36100	100	43240	100	22956	100	29232	100
*g*10	89748	100	179156	100	209228	100	70316	100	193980	100
*g*11	76400	100	76628	100	76400	100	NA	0	322423	92
*g*12	3460	100	3580	100	4546	100	3208	100	2848	100
*g*13	93924	100	83300	100	61579	32	69972	84	NA	0
平均值	49436	99.23	60420	99.23	64263	91.69	38553 + NA	88	83073 + NA	85.54

由表 4 可以看到在成功率上,PJAD-CDE 与 PJAD-CDE1 相同,优于 PJAD-CDE2, PJAD-CDE3, PJAD-CDE4. 在评价次数上 PJAD-CDE 在函数 *g*01-*g*12 以及平均评价次数上均优于 PJAD-CDE1. PJAD-CDE3 在函数 *g*01,*g*02,*g*04 – *g*10,*g*12 的评价次数上均优于其它四种算法,表明使用"DE/rand-to-best/2"策略在收敛速度上有较大优势,但同时易于陷入局部最优导致算法成功率下降. 总体来看 PJAD-CDE 算法相比于使用单个策略的 PJAD-CDE1, PJAD-CDE2, PJAD-CDE3, PJAD-CDE4 算法在保持成功率同时有较好的收敛速度,显示了自适应选择策略的有效性.

7.3 参数自适应设置的有效性验证

PJAD-CDE 算法使用了 JADE 算法机制对参数 CR, *F* 自适应设置. 为了验证参数自适应设置的有效性,将参数 CR,*F* 取文献[5]中的值 CR = 0.9, *F* = 0.5,算法命名为 PJAD-CDE5,算法运行 30 次,函数最大评估次数为 500000 次运行,由运行结果发现 PJAD-CDE 与 PJAD-CDE5 在各函数的算法评价次数上相差不大,即表示参数自适应设置对算法收敛速度影响不明显. 在此仅给出两者在成功率上的对比如表 5,SR 表示成功率.

表 5　PJAD-CDE 与 PJAD-CDE5 在成功率上的对比

函数	*g*01	*g*02	*g*03	*g*04	*g*05	*g*06	*g*07
PJAD-CDE(SR%)	100	90	100	100	100	100	100
PJAD-CDE5(SR%)	100	72	100	100	100	100	100
函数	*g*08	*g*09	*g*10	*g*11	*g*12	*g*13	平均值
PJAD-CDE(SR%)	100	100	100	100	100	100	99.23
PJAD-CDE5(SR%)	100	100	100	0	100	92	89.54

由表 5 可以看到在 PJAD-CDE 算法在函数 *g*02, *g*11,*g*13 以及平均值上的成功率明显优于 PJAD-CDE5. 表明参数 CR,*F* 的自适应设置相比于固定取值方式来说在成功率上作用明显.

8 结束语

本文提出了一种自适应约束差分进化算法,算法提出了一种基于混合种群的自适应适应值的方法来分配信用值并采用概率匹配方式自适应选择差分变异算子,对于参数 CR 与 *F* 采用 JADE 算法机制进行自适应设置. 与其他算法的对比结果验证了算法的有效性,同时也验证了自适应机制的有效性.

参考文献

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

[2] D Thierens. An adaptive pursuit strategy for allocating operator probabilities[A]. Genetic and Evolutionary Computation Conference (GECCO 2005)[C]. Washington DC, USA: ACM Press, 2005. 1539 – 1546.

[3] A Fialho, M Schoenauer, M Sebag, Analysis of adaptive operator selection techniques on the royal road and long k-path problems[A]. Genetic and Evolutionary Computation Conference (GECCO 2009)[C]. Montreal, Canada: ACM Press, 2009. 779 – 786.

[4] Wenyin Gong, Alvaro Fialho, Zhihua Cai, Hui Li. Adaptive strategy selection in differential evolution for numerical optimization: An empirical study[J]. Information Sciences, 2011, 181(24): 5346 – 5386.

[5] Wenyin Gong, A. Fialho, Zhihua Cai, Adaptive strategy selection in differential evolution[A]. Genetic and Evolutionary Computation Conference (GECCO 2010)[C]. Portland, USA: ACM Press, 2010. 409 – 416.

[6] Ke Li, A′ lvaro Fialho, Sam Kwong, Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(1): 114 – 130.

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

[8] Wenyin Gong, Zhihua Cai, Dingwen Liang. Adaptive rank-

ing mutation operator based differential evolution for constrained optimization[J]. IEEE Transactions on Cybernetics,2015,45(4):716 – 727.

[9] Yong Wang,Zixing Cai,Yuren Zhou,An adaptive tradeoff model for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation,2008,12(1):80 – 92.

[10] Runarsson TP,Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation,2000,4(3):284 – 294.

[11] 蔡自兴,江中央,王勇,等. 一种新的基于正交实验设计的约束优化进化算法[J]. 计算机学报,2010,33(5):855 – 864.
Cai Zixing,Jiang Zhongyang,Wang Yong,et al. A novel constrained optimization evolutionary algorithm based on orthogonal experimental design[J]. Chinese Journal of Computers,2010,33(5):855 – 864. (in Chinese)

[12] 郑建国,王翔,刘荣辉,等. 求解约束优化问题的 εDE 算法[J]. 软件学报,2012,23(9):2374 – 2387.
Zheng Jianguo,Wang Xiang,Liu Ronghui,et al. ε-differential evolution algorithm for constrained optimization problems[J]. Journal of Software,2012,23(9):2374 – 2387. (in Chinese)

[13] A W Mohamed,H Z Sabry. Constrained optimization based on modified differential evolution algorithm[J]. Information Sciences,2012,194(1):171 – 208.

[14] 毕晓君,王钰,李博,等. 基于动态迁移的 ε 约束生物地理学优化算法[J]. 计算机研究与发展,2014,51(3):580 – 589.
Bi Xiaojun,Wang Jue,Li Bo,et al. An εconstrained biogeography-based optimization with dynamic migration[J]. Journal of Computer Research and Development,2014,51(3):580 – 589. (in Chinese)

作者简介

閤大海 男,1981 年生于湖北随州. 现为武汉大学计算机学院博士生. 主要研究方向为演化计算,约束优化.
E-mail:xdh628@163.com

李元香 男,1962 年出生于湖北监利,武汉大学计算机学院软件工程国家重点实验室教授,博士生导师,主要研究方向为演化计算的理论与应用研究.
E-mail:yxli@whu.edu.cn

龚文引 男,1979 年出生于湖南永顺,博士,中国地质大学(武汉)计算机学院副教授,主要研究方向为演化计算及应用.
E-mail:wygong@cug.edu.cn