基于旋转学习机制的差分演化算法

刘会超[1,2],吴志健[1]
(1.武汉大学计算机学院 软件工程国家重点实验室,湖北武汉 430072;
2.黄淮学院信息工程学院,河南驻马店 463000)

摘 要: 为克服反向学习机制仅能搜索反向空间中一个固定点的弊端,通过引入旋转操作将其扩展为一种新的旋转学习机制,新机制通过调整旋转角度能搜索旋转空间中的任意一点,具备更强的勘探能力和多种应用模式.通过嵌入旋转学习算子,并引入参数自适应机制,提出了新的基于旋转学习的差分演化算法.在广泛使用的测试函数集上开展仿真实验,结果验证了旋转学习机制的有效性,与多种知名差分演化算法相比,新算法在寻优性能上竞争优势明显,且具有良好的适用性.

关键词: 演化计算; 差分演化; 旋转学习机制; 反向学习机制
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2015)10-2040-07
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.10.024

Differential Evolution Algorithm Using Rotation-Based Learning

LIU Hui-chao[1,2], WU Zhi-jian[1]
(1. *State Key Lab of Software Engineering, Computer School, Wuhan University, Wuhan, Hubei* 430072, *China*;
2. *College of Information Engineering, Huanghuai University, Zhumadian, Henan* 463000, *China*)

Abstract: Opposition-based learning mechanism (OBL) only searches a fixed point in the opposite space. In order to overcome this defect, a rotation-based learning mechanism (RBL) is proposed by introducing the rotation operation to OBL. The RBL can search any point in the rotation space by adjusting the rotation angle parameter, and has a stronger exploration capacity and multiple application modes. By embedding the RBL and self-adaptive parameter control mechanism into differential evolution algorithm (DE), the rotation-based differential evolution algorithm (RDE) is introduced. Simulation experiments conducted on a set of widely used benchmark functions verify the effectiveness of RBL mechanism. Compared with several well-known DE variants, the RDE algorithm has a significant competitive advantage in optimizing performance, and has good applicability.

Key words: evolutionary computation; differential evolution; rotation-based learning; opposition-based learning

1 引言

差分演化算法(Differential Evolution, DE)由 Storn 和 Price[1,2]提出,其在求解复杂优化问题时具有简单、高效和鲁棒性强的优点[3].过去十多年间,DE 一直是演化计算领域的研究热点,众多学者对 DE 进行了大量的研究和改进,并将算法广泛应用于函数优化[4]和各种实际应用领域[5],取得了良好效果.

然而 DE 算法也存在部分缺点,一些算法性能增强算子被引入 DE 以进一步增强算法的性能.由 Tizhoosh[6]提出的反向学习机制(Opposition-Based Learning, OBL)便是其中之一.Rahnamayan 等[7]将 OBL 机制应用于 DE,并用提出的 ODE 算法求解含噪声的优化问题.大规模实验测试表明 OBL 机制可以有效增强 DE 算法的收敛率.但 OBL 机制也存在不足之处,如它只能搜索反向空间中的一个固定的反向点,当优化问题的定义域和值域都对称时,OBL 机制的优势将难以发挥.

在 OBL 机制中,反向数是在一维数轴上计算的.通过扩展 OBL 机制到二维空间,提出了一种新的旋转学习机制(Rotation-Based Learning, RBL).它首先将一个数映射为二维空间一个特定圆上的点,然后将这个点沿圆逆时针旋转指定的角度,得到的新点的 x 轴坐标就是

收稿日期:2014-06-23;修回日期:2014-11-09;责任编辑:李勇锋
基金项目:国家自然科学基金(No.61070008, No.61364025, No.61305150);河南省基础与前沿技术研究计划(No.122300410071);河南省科技攻关计划(No.122102310474)

原数的一个旋转数.通过旋转不同的角度,RBL 机制可以搜索空间中的任意一点,有效克服了 OBL 机制的缺点,且具备多种应用模式.通过将 RBL 算子嵌入到 DE 算法,提出了基于旋转学习的差分演化算法(Rotation-based Differential Evolution,RDE),经分析其时间复杂度和 DE 是一致的.仿真实验表明 RBL 机制可有效提高 DE 的勘探能力,特别适合求解多峰优化问题.与多种知名 DE 算法相比,RDE 在求解精度和收敛速度上都具有明显优势,且在求解高维问题时具有良好适用性.

2　OBL 机制及旋转解释

2.1　OBL 机制简述

OBL 机制[6]的主要思想是同时考查当前解及其反向解,使当前解更易于接近全局最优解.文献[8]证明反向数与随机数相比以更高概率接近全局最优.Rahnamayan 等[9]基于欧氏距离对候选解及其反向解比随机选择的解更易接近全局最优的问题进行了直观的解释和证明.目前,OBL 机制已广泛应用于演化计算领域的算法改进[10]和实际问题求解中[11],具体可参考文献[12].

定义 1　反向数,设实数 z 的取值区间是 $[a,b]$,则 z 的反向数 $\breve{z}$ 定义为:

$$\breve{z} = a + b - z. \tag{1}$$

如图 1 所示,反向数可看作是原数相对于区间 $[a,b]$ 的中心点的镜像.

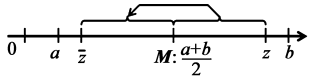

图1　一个数及其反向数的位置关系

定义 2　反向点,设 $\boldsymbol{Z}=(z_1,z_2,\cdots,z_D)$ 是 D 维实空间中的一个点,且 $z_i\in[a_i,b_i]$,$i=1,2,\cdots,D$.则 $\boldsymbol{Z}$ 的反向点 $\breve{\boldsymbol{Z}}=(\breve{z}_1,\breve{z}_2,\cdots,\breve{z}_D)$ 定义为:

$$\breve{z}_i = a_i + b_i - z_i. \tag{2}$$

向量 $\boldsymbol{a}=(a_1,a_2,\cdots,a_D)$ 和 $\boldsymbol{b}=(b_1,b_2,\cdots,b_D)$ 构成了点 $\boldsymbol{Z}$ 的反向空间.显然,给定反向空间和原始点 $\boldsymbol{Z}$,反向点 $\breve{\boldsymbol{Z}}$ 的位置就已确定.

2.2　OBL 的旋转解释

在 OBL 机制中,反向数的计算是在一维数轴上实现的.如果把数轴置于二维平面中,则反向数就可以用旋转的方式来得到.

如图 2 所示,给定由 x 和 y 轴组成的二维平面,数 z 及其取值边界 a 和 $b(a<b)$ 在 x 轴上分别标为点 D、A 和 B.点 C 为区间 $[a,b]$ 的中点,它的坐标是 $\frac{a+b}{2}$.以 C 点为中心,以 $r(=\frac{b-a}{2})$ 为半径画一个圆,过点 D 沿 y 轴正方向作向量与圆交于点 L.显然,数 z 是点 L 的 x 轴坐标,且向量 $\overrightarrow{\boldsymbol{CL}}$ 的模 $|\overrightarrow{\boldsymbol{CL}}|=r$.为计算方便,定义两个中间变量 u 和 v:

$$u = \Pr j_x \overrightarrow{\boldsymbol{CD}} = z - \frac{a+b}{2}, \tag{3}$$

$$v = |\overrightarrow{\boldsymbol{DL}}| = \sqrt{(z-a)\cdot(b-z)},$$

其中 $\Pr j_x \overrightarrow{\boldsymbol{CD}}$ 表示向量 $\overrightarrow{\boldsymbol{CD}}$ 在 x 轴的投影.设交角 $\angle LCB$ 的角度为 α,则有:

$$\cos\alpha = \cos\angle LCB = \frac{\Pr j_x \overrightarrow{\boldsymbol{CD}}}{|\overrightarrow{\boldsymbol{CL}}|} = \frac{u}{r},$$
$$\sin\alpha = \sin\angle LCB = \frac{|\overrightarrow{\boldsymbol{DL}}|}{|\overrightarrow{\boldsymbol{CL}}|} = \frac{v}{r}. \tag{4}$$

让点 L 沿圆逆时针旋转 180 度到达 M 点.点 M 在 x 轴的投影为点 E.显然,点 E 的 x 轴坐标 $\breve{z}$ 就是原数 z 的反向数.因此,OBL 机制中反向数也可解释为原数在二维空间的特定圆上逆时针旋转 180 度所得.该解释也可推广到高维情形.

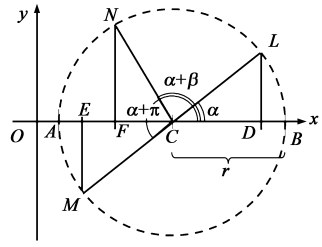

图2　OBL及RBL机制在二维空间中的解释

3　旋转学习机制

3.1　旋转学习的概念

受 2.2 节启发,可通过旋转任意指定的角度将 OBL 机制扩展为新的旋转学习机制 RBL.

定义 3　旋转数,如图 2 所示,如果将点 L 沿圆逆时针旋转 β 度,到达新点 N,则角 $\angle NCB$ 的角度是 $\alpha+\beta$.点 N 在 x 轴的投影点 F 的 x 轴坐标 $\breve{z}$ 称为原数 z 的 β 角旋转数.用 $\breve{u}$ 表示向量 $\overrightarrow{\boldsymbol{CF}}$ 在 x 轴的投影,则有:

$$\begin{aligned}\breve{u} &= \Pr j_x \overrightarrow{\boldsymbol{CF}} = r\times\cos(\alpha+\beta)\\ &= u\times\cos\beta - v\times\sin\beta,\end{aligned} \tag{5}$$

则旋转数 $\breve{z}$ 可表示为:

$$\breve{z} = \left(\frac{a+b}{2}\right) + \breve{u}. \tag{6}$$

旋转数的概念可推广到高维情形,且每一维的旋转数在各自对应的二维空间中独立计算.

定义 4　旋转点,设 $\boldsymbol{Z}=(z_1,z_2,\cdots,z_D)$ 是 D 维空间的一个点,$z_i\in[a_i,b_i]$,$i=1,2,\cdots,D$.类似于式(3),设第 i 维向量 $\overrightarrow{\boldsymbol{C_iD_i}}$ 在 x 轴的投影用 u_i 表示,第 i 维向量 $\overrightarrow{\boldsymbol{D_iL_i}}$ 的模用 v_i 表示,点 $\boldsymbol{Z}$ 各维沿各自对应的圆逆时针旋转 β

度,则点 $\breve{\boldsymbol{Z}}=(\breve{z}_1,\breve{z}_2,\cdots,\breve{z}_D)$称为点 $\boldsymbol{Z}$ 对应的 β 角旋转点,其中:

$$\breve{z}_i=\left(\frac{a_i+b_i}{2}\right)+(u_i\times\cos\beta-v_i\times\sin\beta). \tag{7}$$

显然,OBL 机制得到的都是 180°角旋转点.通过设定不同的旋转角度,RBL 机制可搜索旋转空间中的任意一点,有效克服了 OBL 机制的缺点.

3.2 旋转学习算子

RBL 机制可作为独立算子嵌入基于种群的算法中,以提高算法的寻优性能.给定一个种群,其旋转空间由种群中所有个体各维的当前取值范围确定.算法 1 列出了 RBL 算子的伪代码.算子包括两部分:01)至 04)步是第一部分,用于计算当前种群的旋转空间边界向量、旋转中心向量和旋转半径向量.05)至 09)步是算子的第二部分,用来产生基于旋转的种群.其中每个旋转个体根据式(7)计算得出.旋转角度 β 需要事先设定,可以在[0°,360°]区间内取固定或动态的值.

算法 1 RBL 算子

输入:规模为 NP 的种群 $P=\{X_1,X_2,\cdots,X_{NP}\}$,旋转角度 β.

输出:种群 P 对应的旋转种群 $RP=\{\breve{X}_1,\breve{X}_2,\cdots,\breve{X}_{NP}\}$.

01) for $i=1$ to D do {* 计算种群 P 的旋转空间数据 *}

02) 统计第 i 维的取值边界:$a_i=\min(X_{j,i})$和 $b_i=\max(X_{j,i})$;

03) 计算第 i 维的中心:$c_i=(a_i+b_i)/2$ 和半径:$r_i=(b_i-a_i)/2$;

04) end

05) for $i=1$ to NP do {* 生成旋转种群 RP *}

06) 设置个体 X_i的旋转角度β;

07) 按式(7)对个体 X_i实施旋转,生成旋转个体 $\breve{X}_i$;

08) 将旋转个体 $\breve{X}_i$ 加入旋转种群 RP;

09) end

10) 输出旋转种群 RP;

3.3 应用模式分析

通过调整旋转角度,RBL 算子在搜索时具有很大的灵活性.事实上,RBL 机制存在多种应用模式.

模式 1 当旋转角度固定在 180°时,RBL 机制转化为 OBL 机制.

模式 2 通过精心选择旋转角度,使旋转数正好落于旋转空间中点和反向点之间,则 RBL 机制近似转化为 QODE[13]算法中的准反向学习机制.

模式 3 若用正态分布 $N(\mu,\sigma^2)$来控制旋转角度,则 RBL 将在 μ 角旋转点附近做局部搜索.

模式 4 当旋转角度取值较小(如 ±30°)时,RBL 将在原始点(0 角旋转点)附近做局部搜索.

模式 5 当旋转角度在 0°至 360°之间随机选取时,RBL 机制近似于随机搜索.

易知,RBL 机制既具备传统反向学习机制的一般框架,又可通过调整旋转角度,转化为其他搜索模式,且每种模式具有不同的搜索特点.

4 基于旋转学习的差分演化算法

4.1 RDE 算法框架

关于 DE 算法,众多文献已有详述[2,3],此处略过.本节将把 RBL 算子嵌入 DE 算法,提出基于旋转学习的差分演化算法 RDE.

算法在种群初始化阶段首先随机生成一个基本种群 BP.然后用算法 1 生成 BP 对应的旋转种群 RP.最后,评估这两个种群所有个体的适应值,并选择 NP 个较优个体构成初始种群 P.

在演化阶段,采用旋转概率(probability of rotation,pr ∈(0,1])参数来控制 RBL 机制的执行.在每一演化代,若一个服从 $U(0,1)$的随机数小于参数 pr 的值,则执行 RBL算子来生成当前种群 P 对应的旋转种群 RP,且选择两个种群中 NP 个较优个体作为下一代演化种群;否则,应用标准的 $DE/rand/1/exp$ 演化算子来生成下一代种群.

在 RDE 算法中,旋转概率 pr 和旋转角度 β 是两个新引入的参数.RDE 算法采用 3.3 节的模式 3 来控制参数 β,定义如下:

$$\beta=N(\beta_0,\sigma^2), \tag{8}$$

其中 β_0为旋转角度的基数,σ^2用于控制参数 β 的变化范围,则 RBL 机制将在 β_0角旋转点附近做局部搜索.式(8)同样用于动态调整旋转概率参数 pr,则参数 pr 将在演化过程中自适应变化,使旋转操作非均匀分布于整个演化过程,降低固定参数对算法性能的影响,有利于提高算法的鲁棒性.

算法 2 RDE 算法

输入:问题定义域 S 和目标函数f.

输出:最优解 $Best$.

01) 设置参数 pr_0和 β_0的初始值;

02) 按均匀分布在搜索空间内随机初始化基本种群 BP;

03) 执行算法 1,生成旋转种群 $RP(0)$;

04) 从$\{BP,RP\}$中选择 NP 个较优个体作为初始种群 $P(0)$;

05) while $FES\leqslant MaxFES$ do

06) if rand() $\leqslant pr$ then

07) 执行算法 1,生成当前种群 $P(t)$的旋转种群 $RP(t)$;

08) 从$\{P,RP\}$中选择 NP 个较优个体作为下一代种群 $P(t+1)$;

09) 通过式(8)更新 pr 的值;

10) else

11) 执行经典 $DE/rand/1/exp$ 策略,生成下代种群 $P(t+1)$;

12) end

13) 更新最优个体 $Best$ 的信息;

14) end

15)输出最优个体 $Best$,退出算法.

算法 2 列出了 RDE 算法实现的伪代码.其中 pr_0和 β_0分别表示旋转概率和旋转角度两个参数的初始值.此外,算法 1 中的 06)步需用式(8)代替.

4.2 算法复杂度分析

设待求问题的维度为 D,计算待求问题的时间复杂度为 $O(F)$,种群规模为 NP,最大函数评估次数为 T.容易看出,算法 1 中 01) – 04)步及 05) – 09)步两部分的时间复杂度都是 $O(NP\cdot D)$,即 RBL算子的计算时间复杂度为 $O(NP\cdot D)$.在算法 2 中,02) – 04)步为算法初始化部分,各步的时间复杂度分别是 $O(NP\cdot D\cdot O(F))$、$O(NP\cdot D\cdot O(F))$和 $O(NP\cdot\log(NP))$.05) – 12)步是种群演化阶段,其中 07)、08)及 11)步是关键步骤,各步的时间复杂度分别是 $O(NP\cdot D\cdot O(F))$、$O(NP$

·log(*NP*))和 *O*(*NP*·*D*·*O*(*F*)).去除所有低阶项,并结合算法的循环结构,可得算法 2 的时间复杂度为 *O*(*T*·*D*·*O*(*F*)),这和 DE 算法是一致的,说明引入 RBL 机制并没有增加 DE 算法的时间复杂度.

5　仿真实验与分析

5.1　测试函数与配置

本节选取 13 个广泛使用的测试函数开展仿真实验.其中 F_1 – F_5是单峰函数,F_6是不连续的阶梯函数,F_7是带噪声的函数,F_8 – F_{13}是具有多个局部最优的多峰函数.这些函数的详细描述见文献[14].

RDE 算法中参数 $NP = 60$, $F = 0.5$ 和 $Cr = 0.9$,同时设置 $\beta_0 = 180°$,$\sigma = 45°$,则 β 将在 180°附近波动,并有 95% 的概率落入区间[90°,270°],这既充分利用了 OBL 机制的优势,又可克服其只能搜索固定点的弊端,能够有效扩展搜索范围,提高算法的勘探能力.实验研究显示参数 *pr* 在[0.1,0.3]内取值对演化最有利,故设置参数 $pr_0 = 0.2$, $\sigma = 0.05$. 测试的最大适应值评估次数(*MaxFEs*)为 5000·*D*.算法在完成最大适应值评估次数后停止.每一测试实例均独立运行 30 次,所得结果的均值作为算法性能评估的依据.

为测试 RBL 机制的有效性及 RDE 算法性能,将 RDE 算法和其他 4 个知名 DE 算法 ODE[7]、jDE[15]、SaDE[16]和 JADE[17]进行对比实验,对比算法均使用其原文献中的参数设置.设问题维度 $D = 50$. 表 1 列出了 5 个算法的实验结果,符号 *w*/*t*/*l* 表示 RDE 算法胜过、持平和劣于对比算法的函数个数.

表 1　RDE 算法和其他 4 种算法的求解结果比较($D = 50$)

函数	jDE 均值 ± 方差	JADE 均值 ± 方差	SaDE 均值 ± 方差	ODE 均值 ± 方差	RDE 均值 ± 方差
F_1	3.46E – 18 ± 3.27E – 19	4.90E – 55 ± 3.32E – 55	**2.27E – 69 ± 1.03E – 69**	2.24E – 30 ± 6.25E – 31	1.23E – 30 ± 1.81E – 31
F_2	4.52E – 11 ± 2.56E – 12	7.82E – 31 ± 6.47E – 31	**5.77E – 43 ± 1.39E – 43**	8.64E – 11 ± 1.02E – 11	1.63E – 13 ± 8.35E – 15
F_3	2.45E + 03 ± 1.26E + 02	1.84E + 02 ± 2.03E + 01	**5.33E – 04 ± 4.71E – 04**	3.72E + 01 ± 5.35E + 00	2.34E + 01 ± 1.30E + 00
F_4	**8.04E – 01 ± 9.14E – 02**	5.15E + 00 ± 2.40E – 01	1.13E + 01 ± 3.81E – 01	9.13E – 01 ± 2.65E – 01	1.42E + 00 ± 2.34E – 01
F_5	4.06E + 01 ± 2.01E + 00	5.28E + 01 ± 4.83E + 00	**1.60E + 01 ± 4.47E + 00**	4.58E + 01 ± 1.78E + 00	4.32E + 01 ± 2.07E – 01
F_6	**0.00E + 00 ± 0.00E + 00**	2.40E + 00 ± 4.63E – 01	4.87E + 00 ± 9.94E – 01	3.00E – 01 ± 8.37E – 02	**0.00E + 00 ± 0.00E + 00**
F_7	1.59E – 02 ± 6.75E – 04	6.32E – 03 ± 4.75E – 04	1.04E – 02 ± 4.62E – 04	**2.20E – 03 ± 1.50E – 04**	6.03E – 03 ± 3.21E – 04
F_8	– 16154.80 ± 7.17E + 01	– 20933.40 ± 7.35E + 00	**– 20949.10 ± 6.53E – 13**	– 9264.18 ± 1.76E + 02	**– 20949.10 ± 6.64E – 13**
F_9	9.67E + 01 ± 1.35E + 00	**1.11E – 09 ± 1.10E – 10**	3.32E – 02 ± 3.26E – 02	1.55E + 02 ± 8.89E + 00	7.21E – 05 ± 1.75E – 05
F_{10}	4.58E – 10 ± 2.82E – 11	6.82E – 01 ± 1.28E – 01	1.25E + 00 ± 9.68E – 02	1.50E – 14 ± 7.74E – 16	**9.56E – 15 ± 4.33E – 16**
F_{11}	**0.00E + 00 ± 0.00E + 00**	8.03E – 03 ± 2.18E – 03	6.29E – 03 ± 2.47E – 03	2.79E – 03 ± 8.82E – 04	**0.00E + 00 ± 0.00E + 00**
F_{12}	1.87E – 17 ± 1.10E – 19	8.29E – 03 ± 4.85E – 03	2.90E – 02 ± 1.40E – 02	**1.81E – 17 ± 6.80E – 29**	**1.81E – 17 ± 1.62E – 31**
F_{13}	3.41E – 17 ± 7.57E – 19	1.46E – 03 ± 6.82E – 04	2.43E – 01 ± 1.64E – 01	3.66E – 04 ± 3.60E – 04	**2.88E – 17 ± 2.38E – 31**
w/*t*/*l*	9/2/2	10/0/3	7/1/5	10/1/2	—

5.2　实验结果分析

5.2.1　RBL 机制的有效性分析

从表 1 可知,RDE 算法获得了 6 个测试函数的最优解,其中 3 个为全局最优解,而 ODE 算法只获得了 2 个最优解.而且 RDE 算法在 10 个函数上的求解结果都优于 ODE.这说明 RDE 算法在求解精度上明显优于 ODE 算法,证明 RBL 机制要比 OBL 机制更加有效.

此外,RDE 算法取得了 6 个多峰函数中 5 个的最优解,表明 RDE 算法在求解多峰函数时更有优势.这进一步证明在 RBL 机制中利用高斯分布来生成旋转角度,可扩大 RBL 算子的搜索范围,从而增强 DE 算法的勘探能力,进而提高算法在多峰函数上的寻优效果.

表 2　各对比算法的 Friedman 测试结果($D = 50$)

算法	RDE	jDE	SaDE	ODE	JADE
秩均值	**2.00**	3.15	3.19	3.19	3.46

表 3　RDE 与其他算法的 Wilcoxon 测试结果($D = 50$)

算法	jDE	SaDE	ODE	JADE
显著性	6.15E – 02	1.70E – 01	**2.61E – 02**	**1.71E – 03**

5.2.2　RDE 算法性能分析

由表 1 底部的对比结果可知 RDE 算法分别在 9、10、7 和 10 个函数上优于 jDE、JADE、SaDE 和 ODE 算法,说明 RDE 在求解精度上明显好于其他算法.利用 SPSS 软件对 5 种算法的求解结果进行非参数检验.表 2 列出

了各算法的 Friedman 测试结果,显然 RDE 算法排名第一,这表明 RDE 算法的综合性能是最好的.表 3 是 RDE 与其他算法的 Wilcoxon 测试结果,显著性水平为 0.05.可以看出 RDE 明显优于 ODE 和 JADE 算法.尽管 RDE 与 jDE 和 SaDE 算法的统计优势并不明显,但根据表 2,RDE 算法的综合性能仍然较优.

图 3 展示了所有 5 个算法在函数 F_6、F_{10}、F_{11}和 F_{13} 上的收敛曲线.一般认为搜索能力强的算法其收敛速度可能会较慢,但从图 3 可知 RDE 算法的收敛速度依然比较快.

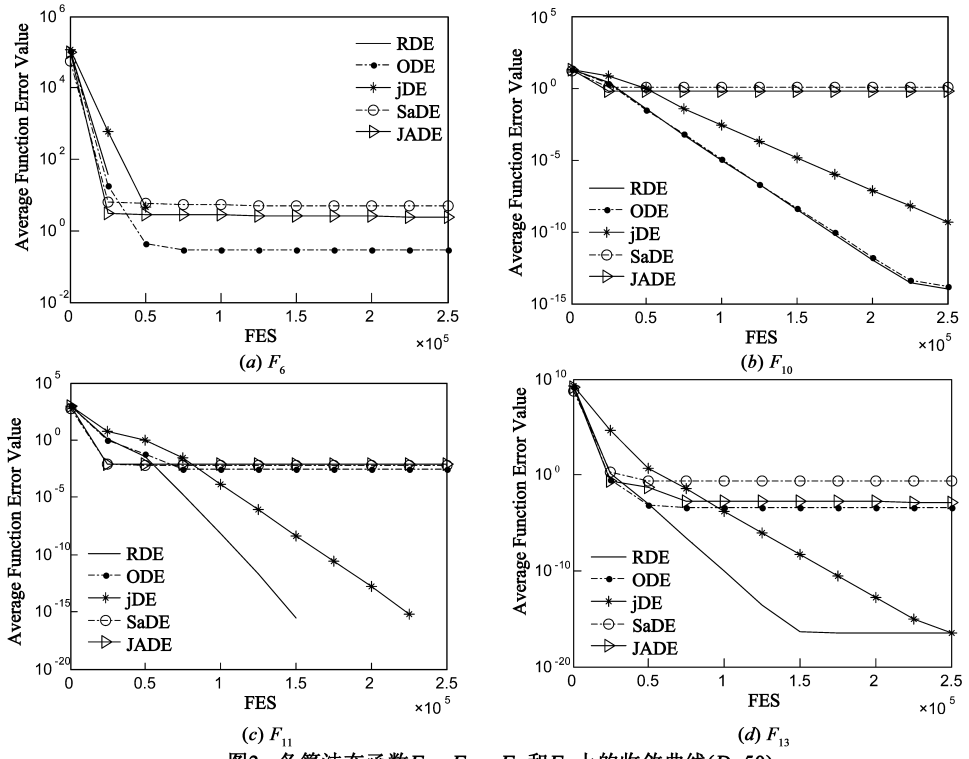

图3 各算法在函数F_6、F_{10}、F_{11}和F_{13}上的收敛曲线(D=50)

表 4 RDE 和其他算法的求解结果比较(D = 100)

函数	jDE 均值 ± 方差	JADE 均值 ± 方差	SaDE 均值 ± 方差	ODE 均值 ± 方差	RDE 均值 ± 方差
F_1	1.56E − 20 ± 1.66E − 21	9.56E − 22 ± 4.30E − 22	**6.74E − 42 ± 1.53E − 42**	5.93E − 24 ± 3.75E − 24	3.51E − 29 ± 6.81E − 30
F_2	**0.00E + 00 ± 0.00E + 00**	7.00E − 01 ± 1.77E − 01	1.33E − 01 ± 6.21E − 02	3.00E − 01 ± 1.50E − 01	**0.00E + 00 ± 0.00E + 00**
F_3	1.43E + 04 ± 6.08E + 02	4.84E + 03 ± 2.60E + 02	**4.38E − 01 ± 1.94E − 01**	1.87E + 03 ± 1.35E + 02	1.63E + 03 ± 4.75E + 01
F_4	2.25E + 01 ± 4.89E − 01	1.25E + 01 ± 2.09E − 01	1.19E + 01 ± 2.48E − 01	**4.37E + 00 ± 3.04E − 01**	1.29E + 01 ± 6.94E − 01
F_5	1.05E + 02 ± 5.89E + 00	2.18E + 02 ± 1.35E + 01	1.10E + 02 ± 6.88E + 00	1.35E + 02 ± 8.33E + 00	**9.44E + 01 ± 1.44E + 00**
F_6	**0.00E + 00 ± 0.00E + 00**	1.27E + 02 ± 1.38E + 01	1.03E + 01 ± 1.45E + 00	2.13E + 01 ± 5.58E + 00	**0.00E + 00 ± 0.00E + 00**
F_7	2.48E − 02 ± 6.83E − 04	6.22E − 02 ± 5.61E − 03	2.75E − 02 ± 1.17E − 03	2.16E − 02 ± 1.74E − 03	**1.11E − 02 ± 6.24E − 04**
F_8	− 2.69E + 04 ± 1.14E + 02	− 4.19E + 04 ± 2.07E + 01	− 4.20E + 04 ± 2.48E − 01	− 1.71E + 04 ± 4.13E + 02	**− 4.20E + 04 ± 2.05E − 01**
F_9	3.10E + 02 ± 2.55E + 00	**6.74E − 09 ± 8.56E − 10**	8.33E − 01 ± 5.43E − 01	1.19E + 02 ± 2.55E + 01	2.41E − 04 ± 6.66E − 05
F_{10}	1.27E − 01 ± 1.25E − 01	3.75E + 00 ± 8.03E − 02	1.79E + 00 ± 6.24E − 02	4.87E − 01 ± 9.63E − 02	**1.38E − 14 ± 2.13E − 16**
F_{11}	**5.78E − 20 ± 2.47E − 21**	3.91E − 02 ± 1.11E − 02	4.41E − 03 ± 2.48E − 03	1.51E − 02 ± 7.72E − 03	6.87E − 20 ± 4.38E − 21
F_{12}	9.06E − 18 ± 7.81E − 22	1.27E − 01 ± 4.01E − 02	1.77E − 02 ± 8.66E − 03	1.04E − 03 ± 1.02E − 03	**9.05E − 18 ± 3.69E − 31**
F_{13}	**2.88E − 17 ± 1.18E − 20**	1.88E + 00 ± 5.38E − 01	1.07E − 01 ± 7.27E − 02	1.46E − 02 ± 7.14E − 03	**2.88E − 17 ± 1.58E − 30**
$w/t/l$	10/2/1	11/0/2	10/0/3	12/0/1	—

5.3 算法适用性分析

表4展示了5种算法在维度 $D=100$ 时的求解结果.RDE算法在函数 F_1、F_5-F_8、F_{10}及 $F_{12}-F_{13}$上均取得了最优值.表4底部的对比结果表明在求解精度上RDE在5种算法中具有绝对优势.表5列出了各对比算法的Firedman测试排名,易知RDE算法的综合性能仍是最优的.表6展示了RDE和其他算法的Wilcoxon测试结果,置信水平0.05.可以看出RDE明显优于jDE、ODE和JADE算法.尽管RDE与SaDE的差异并不明显,但这主要是因为SaDE的求解结果也普遍好于其他3个算法所致.由表4可知,RDE在10个函数上的求解精度优于SaDE算法.

综上表明,RDE算法在求解高维问题时依然性能优异,说明RDE算法的适用性非常好,可以求解不同类别和难度的优化问题.这也进一步验证了RBL机制的有效性和RDE算法的寻优能力.

表5　各对比算法的Friedman测试结果($D=100$)

算法	RDE	SaDE	jDE	ODE	JADE
秩均值	**1.69**	2.81	2.96	3.31	4.23

表6　RDE与其他算法的Wilcoxon测试结果($D=100$)

算法	SaDE	jDE	ODE	JADE
显著性	8.81E-02	**2.93E-03**	**3.05E-03**	**3.05E-03**

6 结论

通过扩展OBL机制提出了一种新的旋转学习机制RBL,它可以搜索旋转空间中的任意一个点,具备更大的搜索范围,且应用模式多样.通过嵌入RBL算子,提出了基于旋转学习的差分演化算法RDE,其计算时间复杂度与DE算法一致.仿真实验将RDE和其他知名DE算法进行对比.结果证明RBL机制可有效提高算法的勘探能力,尤其适合求解多峰优化问题,而且RDE算法是一种非常有竞争力的新算法,具有良好的适用性.未来将进一步研究RBL机制的搜索特点,并应用到其他智能算法中.

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, CA: International Computer Science Institute, 1995. TR-95-012.

[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] Das S, Suganthan P N. Differential evolution: a survey of the state-of-the-art [J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 4-31.

[4] Wang H, Rahnamayan S, Sun H, et al. Gaussian bare-bones differential evolution [J]. IEEE Transactions on Cybernetics, 2013, 43(2): 634-647.

[5] 王则林,吴志健,尹兰. IPV6环境下的高维大规模包匹配算法[J].电子学报,2013,41(11):2181-2186.
Wang Zelin, Wu Zhijian, Yin Lan. High-dimension large-scale packet matching algorithm in IPV6 [J]. Acta Electronica Sinica, 2013, 41(11): 2181-2186. (in Chinese)

[6] Tizhoosh H R. Opposition-based learning: a new scheme for machine intelligence [A]. International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce [C]. Vienna, Austria: IEEE Press, 2005. 695-701.

[7] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-based differential evolution [J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 64-79.

[8] Rahnamayan S, Tizhoosh H R, Salama M. Opposition versus randomness in soft computing techniques [J]. Applied Soft Computing, 2008, 8(2): 906-918.

[9] Rahnamayan S, Wang G G, Ventresca M. An intuitive distance-based explanation of opposition-based sampling [J]. Applied Soft Computing, 2012, 12(9): 2828-2839.

[10] 周新宇,吴志健,王晖,等.一种精英反向学习的粒子群优化算法[J].电子学报,2013,41(8):1647-1652.
Zhou Xinyu, Wu Zhijian, Wang Hui, et al. Elite opposition-based particle swarm optimization [J]. Acta Electronica Sinica, 2013, 41(8): 1647-1652. (in Chinese)

[11] Shaw B, Mukherjee V, Ghoshal S P. A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems [J]. International Journal of Electrical Power & Energy Systems, 2012, 35(1): 21-33.

[12] Al-Qunaieer F S, Tizhoosh H R, Rahnamayan S. Opposition based computing—a survey [A]. The 2010 International Joint Conference on Neural Networks (IJCNN'10) [C]. Barcelona, Spain: IEEE Press, 2010. 1-7.

[13] Rahnamayan S, Tizhoosh H R, Salama M M A. Quasi-oppositional differential evolution [A]. IEEE Congress on Evolutionary Computation (CEC'07) [C]. Singapore: IEEE Press, 2007. 2229-2236.

[14] Yao X, Liu Y, Lin G. Evolutionary programming made faster [J]. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82-102.

[15] 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.

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

[17] Zhang J, Sanderson A C. JADE: self-adaptive differential evolution with fast and reliable convergence performance[A]. IEEE Congress on Evolutionary Computation (CEC'07)[C]. Singapore: IEEE Press, 2007. 2251 – 2258.

作者简介

刘会超 男, 1982 年生于河南省确山县. 武汉大学计算机学院博士研究生, 研究方向为智能计算.
E-mail: huichaoliu@whu.edu.cn

吴志健(通信作者) 男, 1963 年生, 教授、博士生导师, 武汉大学软件工程国家重点实验室副主任, 研究方向为智能计算、并行计算和智能信息处理.
E-mail: zhijianwu@whu.edu.cn