文章编号: 1001-0920(2017)10-1775-07 DOI: 10.13195/j.kzyjc.2016.1028

融合均匀变异与高斯变异的蝙蝠优化算法

李 煜[1], 裴宇航[2], 刘景森[3†]

(1. 河南大学 管理科学与工程研究所, 河南 开封 475004; 2. 河南大学 计算机与信息工程学院, 河南 开封 475004; 3. 河南大学 智能网络系统研究所, 河南 开封 475004)

摘 要: 为提高蝙蝠算法的寻优精度和收敛速度,提出一种融合均匀变异和高斯变异的蝙蝠优化算法. 算法引入变异开关函数,该函数使所有蝙蝠个体在任何时期都有概率发生变异,使种群保持较高的多样性和活跃性. 同时在算法整个寻优过程中融入均匀变异和高斯变异,两种变异机制共同协作使算法首先快速定位到全局最优解区域,随后完成局部精确搜索. 仿真结果表明,改进后的算法寻优性能显著提高,具有较快的收敛速度和较高的收敛精度.

关键词: 蝙蝠算法; 均匀变异; 高斯变异; 变异开关函数

中图分类号: TP301.6 **文献标志码:** A

Bat optimal algorithm combined uniform mutation with Gaussian mutation

LI Yu[1], *PEI Yu-hang*[2], *LIU Jing-sen*[3†]

(1. Research Institute of Management Science and Engineering, He'nan University, Kaifeng 475004, China; 2. College of Computer and Information Engineering, He'nan University, Kaifeng 475004, China; 3. Institute of Intelligent Network System, He'nan University, Kaifeng 475004, China)

Abstract: In order to improve the convergence speed and the precision of bat algorithm(BA), a bat optimal algorithm combined uniform mutation with Gaussian mutation(UGBA) is proposed. A mutation switch function which makes all individuals have the probability of mutation operation in the whole process is introduced to ensure the population diversity and activity. Meanwhile, uniform mutation and Gaussian mutation are integrated into the whole procedure, which can allow the algorithm to locate global optimal space quickly and implement the accuracy of the solution. Simulation results show that the improved algorithm has faster convergence speed and higher convergence accuracy.

Keywords: bat algorithm; uniform mutation; Gaussian mutation; mutation switch function

0 引 言

传统的数值计算方法在求解越来越复杂的优化问题时,暴露出很多自身缺陷,为此许多研究人员开始探寻新的求解算法. 他们受到自然界中不同生物的启发,提出了很多启发式智能算法,如,受鸟群觅食行为启发提出的粒子群算法[1-2]、受到蚁群在寻找食物过程中发现路径行为启发提出的蚁群算法[3]、受生物进化演变规律启发提出的遗传算法[4]、受果蝇觅食行为启发提出的果蝇优化算法[5]、受青蛙群体寻找食物移动规律启发提出的混合蛙跳算法[6]等. 这些启发式算法在解决各类复杂优化问题上都能提供比传统数值计算方法更好的解.

2010年, Yang[7]提出了一种新的元启发式算法——蝙蝠算法. 蝙蝠算法是模拟蝙蝠发出和接收自身发出的超声波来捕食猎物而提出的一种全局型智能优化算法. 该算法中,每只蝙蝠在搜索空间中的位置代表一个解,对于不同的适应度函数,每只蝙蝠都有自己的适应度值,算法通过比较每只蝙蝠的适应度值来找出当前全局最优位置,然后调整蝙蝠种群的频率、脉冲发射率、响度,朝着当前最优位置不断搜索,最终找到全局最优解. 目前,该算法已经成功用于解决经济调度问题[8]、PMSM(永磁同步电机)参数识别问题[9]、图像压缩问题[10]、无线传感器网络的划分问题[11]、路径规划等问题[12].

收稿日期: 2016-08-12; **修回日期:** 2017-01-15.

基金项目: 河南省科技攻关项目(162102110109); 河南省科技攻关重点项目(142102210036).

作者简介: 李煜(1969–), 女, 教授, 博士, 从事智能算法、电子商务等研究; 裴宇航(1990–), 男, 硕士生, 从事智能算法的研究.

†通讯作者. E-mail: ljs@henu.edu.cn

虽然蝙蝠算法有着模型简单、算法参数少、通用性强等优点,但蝙蝠算法也存在易陷入局部极值、收敛精度不高、算法后期收敛速度慢的缺点. 针对这些不足,国内外许多学者对蝙蝠算法作出了相应的改进,如文献[13]将蝙蝠种群分为外部子种群和内部子种群,外部子种群采用动态惯性权重模型更新蝙蝠速度,内部子种群采用莱维飞行模型更新蝙蝠速度,从而提高了种群全局搜索能力和局部搜索能力,并且保持了种群的多样性. 文献[14]使用随机蝙蝠引导个体飞行和局部搜索,并使用修改部分维的策略提高了种群的多样性,加强了算法的局部求精能力. 文献[15]将模拟退火算法和高斯扰动融入基本蝙蝠算法,使算法不仅具有较好的全局搜索能力,而且加快了算法的收敛速度. 文献[16]在速度公式第1项设置了自适应惯性权重,使蝙蝠在搜索过程中能动态地调整速度. 文献[17]将差分进化算法中的变异、交叉、选择机制应用于蝙蝠算法,使蝙蝠算法具有变异机制,增强了算法的寻优能力.

针对基本蝙蝠算法的不足,本文提出一种融合均匀变异与高斯变异的蝙蝠优化算法(UGBA). 改进后的算法首先利用变异开关函数判定每一代的蝙蝠个体是否进行变异操作;然后对需要变异的蝙蝠个体进行均匀变异或高斯变异,以提高算法的寻优精度和收敛速度;最后对不同维数的经典测试函数进行了测试. 仿真结果表明,改进后算法的寻优精度和收敛速度有较大提升.

1 基本蝙蝠算法

蝙蝠的种类很多,大小各不相同,其中小型蝙蝠发出超声波并且接受其回声来探寻周围的猎物. 蝙蝠发射的声波频率在25～150 kHz之间,波长在2～14 mm之间,这样的波长符合蝙蝠猎物的大小. 声波响度能够达到110 dB,每个声波持续的时间非常短,在5～20 ms之间. 小型蝙蝠大约每秒发射出10～20个超声波. 蝙蝠算法模拟回声定位[18]建立三维场景,算法的更新迭代次数为t,每只蝙蝠能够在位置x_i以频率f_i发出响度为A_0的超声波搜索猎物,以速度v_i向猎物飞行,并实时根据猎物与自己的距离调整自己的飞行速度v_i、脉冲响度A_0和脉冲发射率r_i,逐渐向猎物靠近,最终成功捕食猎物. 算法中蝙蝠的频率、速度和位置在第t代的更新公式如下:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \tag{2}$$

$$x_i^t = x_i^{t-1} + v_i^t. \tag{3}$$

其中:f_i为第i只蝙蝠发出的频率;$f_{\min}$和$f_{\max}$为蝙蝠发出频率的最小值和最大值;β为在$[0,1]$均匀分布的随机向量;$f_i \in [f_{\min}, f_{\max}]$;$x_i$和$v_i$分别为搜索空间中第$i$只蝙蝠的位置和速度,$i = 1,2,\cdots,N$;$x_*$为当前全局的最优解.

在算法逐渐靠近全局最优解时,采用局部搜索,局部位置更新公式如下:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t. \tag{4}$$

其中:ε为$[-1,1]$上的一个随机数[7],A^t为这一代所有蝙蝠响度的平均值.

蝙蝠发出的脉冲响度A_i^t和脉冲发射速率r_i随着迭代进行更新,脉冲响度逐渐减低的同时会提高脉冲发射速率,更新公式如下:

$$A_i^{t+1} = \alpha A_i^t, \tag{5}$$

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)]. \tag{6}$$

其中:α和γ为常数,$0 < \alpha < 1, \gamma > 0$.

2 改进后的蝙蝠算法

2.1 变异开关函数

分析基本蝙蝠算法的6个基本公式发现,算法前期收敛速度很快,但随着迭代次数的增加,蝙蝠越来越靠近全局最优解,种群聚集现象在算法后期比较严重,蝙蝠的速度逐渐减小并趋于0,这导致算法后期易陷入局部极值而停滞不前,降低了算法整体的收敛速度和寻优精度,这种现象是由于蝙蝠算法缺乏避免陷入局部极值的机制和有效跳出局部极值的方法所造成的. 为了解决这个问题,本文在基本蝙蝠算法中融入了均匀变异和高斯变异两种变异机制[19],旨在提高算法的寻优精度和收敛速度,并设置了变异开关函数. 算法在运行过程中,如果满足相应条件,则对当前蝙蝠位置进行变异操作. 变异开关函数表达式如下:

$$\text{switch}(i) = \varphi + \mu * \frac{\mathrm{e}^{[\frac{n(i-1)}{N}]} - 1}{\mathrm{e}^n - 1} + \text{rand} - 1. \tag{7}$$

其中:n为函数的维度,N为蝙蝠种群规模. 如果$0 < \text{switch}(i) < 1$,则对第i只蝙蝠进行变异操作. 经仿真发现,当$\varphi \in [0.4, 0.6], \mu \in [0.1, 0.5]$时,种群中约有三分之二的蝙蝠会发生变异,此时算法的寻优性能最佳. 在算法的整个过程中,每只蝙蝠都有机会发生变异,这种机制打破了传统的只在算法后期对全局最优个体进行变异[20]的界限.

为了验证switch(i)是否影响算法的性能,对比分析原有算法与删除switch(i)的算法(WS),分别运行5个测试函数(维数$n = 30$),对比结果如表1所示. UGBA与WS运行$f_1(x)$的收敛曲线如图1所示.

表 1　UGBA与WS的仿真结果

函数算法	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
UGBA	5.661e-90	5.215e-15	0	2.224e-20	0
WS	6.318e-15	2.935e-07	6.215e-12	7.917e-17	7.917e-17

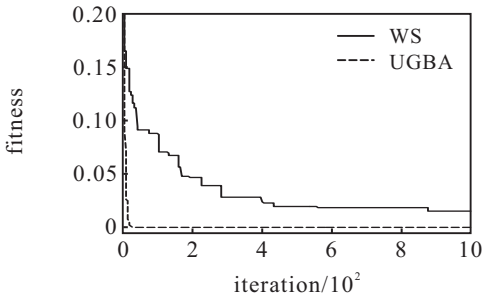

图 1　UGBA与WS运行$f_1(x)$的收敛曲线

由表1和图1可见,删除switch(i)后,算法的收敛精度和收敛速度大幅下降,算法陷入局部最优解的次数明显增多,表明了switch(i)能够提高算法的寻优性能.

2.2　变异过程分析

在算法前期,蝙蝠种群中有一部分蝙蝠个体满足变异条件,会利用均匀变异大步向最优解区域移动,其余蝙蝠继续按照原来的移动方式向最优解区域靠近. 这两种移动方式的结合,加快了种群搜寻最优解区域的速度,缩短了前期搜索的迭代次数并减少了算法陷入局部极值的次数. 利用均匀变异移动的位置更新公式如下:

$$x_i(t) = (1+\tau)x_i(t),\ \mathrm{rand} \leqslant \left(1 - \frac{t}{T_{\max}}\right), \quad (8)$$

其中τ为$[0,1]$上服从均匀分布的随机变量. 在算法后期,对满足变异条件的蝙蝠个体进行高斯变异. 高斯分布的概率密度函数图像关于期望对称,函数曲线下约99.8 %的面积在期望左右3倍标准差的范围内,即所得的随机数集中在以期望为中心的局部区域. 因此,高斯分布变异可以使蝙蝠个体在当前最优解附近产生小幅度的变异,在寻优后期,使算法能够跳出局部极值,提高了算法的寻优精度. 本文根据高斯分布概率密度函数构造了高斯变异函数$\mathrm{GMF}_i(t)$,具体函数表达式如下:

$$\mathrm{GMF}_i(t) = \frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{(-\frac{(x_*-x_i^t)^2}{2\sigma^2})}. \quad (9)$$

其中: x_*为当前全局最优解,σ为高斯分布标准差. 在算法后期,x_*像磁铁一样,将整个蝙蝠种群都吸向自己,造成算法易陷入局部极值,这时通过$\mathrm{GMF}_i(t)$的扰动,种群能够脱离x_*的吸引,减少了陷入局部极值的次数. 利用高斯变异移动的位置更新公式如下:

$$x_i(t) = \mathrm{GMF}_i(t)x_i(t),\ \mathrm{rand} > \left(1 - \frac{t}{T_{\max}}\right). \quad (10)$$

由式(8)和(10)可以看出,均匀变异和高斯变异互相配合协作,不仅加快了算法向最优解区域靠近的速度,而且提高了算法跳出局部极值的能力和寻优精度,因此提高了算法的寻优性能.

2.3　算法流程

Step 1: 随机初始化蝙蝠种群$x_i(i=1,2,\cdots,N)$、蝙蝠的速度v_i、蝙蝠发出的响度A_i^t和脉冲发射速率r_i.

Step 2: 由目标函数$f(x)$, $x=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$,求出每只蝙蝠的目标函数值,并找出当前最优值. 记录最优值蝙蝠的位置为x_*.

Step 3: 根据式(7)判断蝙蝠个体是否变异,如果switch$(i)<0$,则转至Step 4,否则转至Step 5.

Step 4: 根据式(8)～(10)进行变异操作.

Step 5: 蝙蝠个体根据式(2)和(3)更新其速度和位置.

Step 6: 将通过Step 4和Step 5更新出来的蝙蝠个体x_i代入目标函数并求其函数值,如果$f(x_i) < f(x_*)$,则x_i记作当前最优解x_*.

Step 7: 产生一个随机数rand,如果rand$> r_i$,则利用式(4)在最优解附近产生一个局部新解x_{new}.

Step 8: 产生一个随机数,若rand$<$ A_i且$f(x_{\mathrm{new}}) < f(x_*)$,则将Step 7产生的新解记为当前最优解$x_*$,然后按照式(5)和(6)更新响度$A_i^t$和脉冲发射速率$r_i$.

Step 9: 对所有蝙蝠的适应度值排序,找出当前最优解x_*.

Step 10: 重复Step 3～Step 9,直到求出满足精度要求的解或算法达到设定的最大迭代次数.

Step 11: 输出全局最优解.

3　数值仿真与分析

为了测试改进后算法的寻优性能,选取基本蝙蝠算法(BA)[21]、DLBA[22]和IBA[23]与本文提出的融合均匀变异和高斯变异的蝙蝠算法(UGBA),通过9个经典测试函数[24-25]进行对比仿真,具体测试函数如下:

1) Sphere函数

$$f_1(x) = \sum_{i=1}^{n} x_i^2.$$

该函数在$(0,\cdots,0)$处取得最小值0.

2) Schwefel's problem 2.22函数

$$f_2(x) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i|.$$

该函数在$(0,\cdots,0)$处取得最优值0.

3) Griewank函数

$$f_3(x)=1+\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\Big(\frac{x_i}{\sqrt{i}}\Big).$$

该函数在$(0,\cdots,0)$处取得最小值0.

4) Ackley's函数

$$f_6(x)=20+e-20\exp\Big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\Big)-\exp\Big(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\Big).$$

该函数在$(0,\cdots,0)$处取得最小值0.

5) Rastrigin函数

$$f_5(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10].$$

该函数在$(0,\cdots,0)$处取得最小值0.

6) Ronsenbrock函数

$$f_6(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2].$$

该函数在$(0,\cdots,0)$处取得最小值0.

7) Booth's函数

$$f_7(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2.$$

该函数在$(1,3)$处取得最小值0.

8) Easom's函数

$$f_8(x)=-\cos(x_1)\cos(x_2)\exp[-((x_1-\pi)^2+(x_2-\pi)^2)].$$

该函数在(π,π)处取得最小值-1.

9) Six-Hump Camel-Back函数

$$f_9(x)=4x_1^2-2.1x_1^4+\frac{x_1^6}{3}+x_1x_2-4x_2^2+4x_2^4.$$

该函数在$(-0.089\,8,0.712\,6)$和$(0.089\,8,-0.712\,6)$处取得最小值$-1.031\,6$.

在这9个测试函数中,$f_1(x)$、$f_2(x)$、$f_8(x)$和$f_9(x)$为单峰函数,其中$f_2(x)$的全局最优解被局部极值环绕,很难找到最优解. $f_3(x)$、$f_4(x)$、$f_5(x)$、$f_6(x)$和$f_9(x)$为多峰函数,其中$f_3(x)$和$f_4(x)$有很多局部极值,且$f_5(x)$的局部极值数量随着维数的增加而增加. 因此,这9个测试函数都有一定的求解难度,适合测试算法的寻优性能.

3.1 寻优精度分析

为了体现本文算法在问题维数和复杂度上的优越性,实验将函数的维数分别设置为$n=10,30,50,100$. 4种算法在不同维数下运行9个测试函数,参数设置与基本蝙蝠算法[21](BA)保持一致,保证了实验的公平和客观,即种群规模为$N=40$,声波响度$A_0=0.25$,脉冲发射速率$r_0=0.5$. 4种算法的最大迭代次数均设置为1 000代,表2~表8分别统计了不同维数下的最差解、最优解和平均值,并加粗了精度最高的解.

表2 4种算法对$f_1(x)$的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	5.903e-07	1.732e-07	3.290e-07
	DLBA	3.157e-20	2.030e-20	2.614e-20
	IBA	4.567e-21	5.217e-22	1.029e-21
	UGBA	**0**	**0**	**0**
30	BA	8.069e-06	3.418e-06	4.901e-06
	DLBA	5.597e-18	1.957e-18	3.669e-18
	IBA	5.997e-19	4.234e-19	4.579e-19
	UGBA	**8.375e-90**	**3.657e-90**	**5.661e-90**
50	BA	2.074e-05	1.076e-05	1.460e-05
	DLBA	3.914e-16	1.021e-16	2.091e-16
	IBA	7.917e-17	5.267e-17	6.914e-17
	UGBA	**9.411e-86**	**6.197e-86**	**8.232e-86**
100	BA	7.663e-04	3.680e-04	5.060e-04
	DLBA	4.319e-14	2.034e-14	3.109e-14
	IBA	5.391e-15	1.009e-15	4.134e-15
	UGBA	**3.253e-80**	**2.158e-81**	**8.341e-81**

表3 4种算法对$f_2(x)$的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	0.002851	0.001016	0.001743
	DLBA	7.834e-15	5.792e-15	6.797e-15
	IBA	6.637e-15	2.167e-15	3.457e-15
	UGBA	**5.035e-20**	**2.413e-20**	**3.495e-20**
30	BA	0.079594	0.012655	0.072501
	DLBA	5.121e-13	3.114e-13	4.015e-13
	IBA	2.171e-13	8.124e-14	9.491e-14
	UGBA	**8.526e-15**	**3.679e-15**	**5.215e-15**
50	BA	2.557401	0.647940	1.540100
	DLBA	5.214e-10	1.281e-10	3.994e-10
	IBA	2.147e-10	2.014e-10	2.013e-10
	UGBA	**3.021e-12**	**1.512e-12**	**2.236e-12**
100	BA	4.309401	2.804021	3.560981
	DLBA	9.124e-08	6.249e-08	7.183e-08
	IBA	5.215e-08	1.027e-08	3.019e-08
	UGBA	**7.024e-10**	**4.697e-10**	**5.516e-10**

表 4　4种算法对 $f_3(x)$ 的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	3.561e-08	1.703e-08	2.501e-08
	DLBA	6.840e-15	2.172e-15	4.934e-15
	IBA	7.101e-15	5.631e-15	6.251e-15
	UGBA	**0**	**0**	**0**
30	BA	7.421e-07	1.368e-07	3.201e-07
	DLBA	3.471e-14	1.204e-14	2.019e-14
	IBA	4.521e-14	2.049e-14	3.343e-14
	UGBA	**0**	**0**	**0**
50	BA	1.117e-06	3.559e-07	5.871e-07
	DLBA	7.154e-11	2.114e-11	4.919e-11
	IBA	5.104e-12	2.242e-12	3.661e-12
	UGBA	**0**	**0**	**0**
100	BA	3.306e-06	1.283e-06	2.312e-06
	DLBA	3.112e-09	8.179e-10	1.091e-09
	IBA	4.121e-10	3.142e-10	3.317e-10
	UGBA	**0**	**0**	**0**

表 5　4种算法对 $f_4(x)$ 的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	1.868e-05	8.312e-05	1.501e-05
	DLBA	7.214e-19	2.157e-19	4.527e-19
	IBA	4.354e-20	2.927e-20	3.561e-20
	UGBA	**5.817e-22**	**3.472e-22**	**4.537e-22**
30	BA	5.131e-05	3.201e-05	2.831e-05
	DLBA	6.147e-17	2.891e-17	4.617e-17
	IBA	5.617e-18	2.927e-18	4.657e-18
	UGBA	**4.524e-20**	**2.224e-20**	**3.312e-20**
50	BA	3.314e-04	1.413e-04	2.330e-04
	DLBA	8.147e-14	5.781e-14	6.149e-14
	IBA	5.279e-14	2.472e-14	3.671e-14
	UGBA	**6.452e-16**	**2.151e-16**	**4.025e-16**
100	BA	3.845e-03	1.540e-03	2.671e-03
	DLBA	4.172e-11	3.427e-11	3.661e-11
	IBA	9.141e-12	5.237e-12	6.631e-12
	UGBA	**7.985e-12**	**4.316e-12**	**5.253e-12**

表 6　4种算法对 $f_5(x)$ 的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	1.972e-10	3.553e-11	6.790e-10
	DLBA	9.157e-14	5.148e-14	7.541e-14
	IBA	5.781e-15	3.678e-15	4.497e-15
	UGBA	**0**	**0**	**0**
30	BA	5.687e-08	2.417e-08	3.678e-08
	DLBA	7.915e-12	3.918e-12	5.678e-12
	IBA	5.913e-13	2.914e-13	4.289e-13
	UGBA	**0**	**0**	**0**
50	BA	5.218e-06	2.189e-06	4.827e-06
	DLBA	7.934e-10	2.394e-10	5.917e-10
	IBA	5.694e-11	2.891e-11	4.957e-11
	UGBA	**0**	**0**	**0**
100	BA	5.264e-04	3.217e-04	4.251e-04
	DLBA	4.165e-08	2.657e-08	3.457e-08
	IBA	5.921e-09	2.924e-09	4.657e-09
	UGBA	**0**	**0**	**0**

表 7　4种算法对 $f_6(x)$ 的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	8.621e-08	9.645e-09	5.264e-08
	DLBA	9.818e-12	3.424e-12	6.868e-12
	IBA	5.799e-13	8.795e-14	3.125e-13
	UGBA	**9.425e-15**	**5.678e-15**	**6.969e-15**
30	BA	2.294e-06	2.502e-07	1.630e-06
	DLBA	4.326e-10	1.234e-10	2.315e-10
	IBA	9.120e-11	2.196e-11	4.098e-11
	UGBA	**4.149e-13**	**1.754e-13**	**2.537e-13**
50	BA	6.124e-04	2.516e-04	4.159e-04
	DLBA	5.681e-08	2.597e-08	3.645e-08
	IBA	5.627e-09	2.768e-09	4.016e-09
	UGBA	**5.391e-12**	**2.991e-12**	**3.617e-12**
100	BA	0.081567	0.035925	0.56912
	DLBA	4.159e-06	2.036e-06	3.624e-06
	IBA	5.128e-07	2.697e-07	3.691e-07
	UGBA	**9.427e-10**	**5.291e-10**	**7.549e-10**

由表2～表8可见,在相同维数下,UGBA的寻优精度明显高于BA、DLBA和IBA. 在不同维数下,对于函数 $f_1(x)$、$f_2(x)$、$f_4(x)$、$f_6(x)$ 和 $f_7(x)$,BA、DLBA、IBA和UGBA的寻优精度随着维数的增加而降低,但UGBA的收敛精度均明显高于BA、DLBA和IBA. 对于函数 $f_3(x)$ 和 $f_5(x)$,UGBA的寻优精度并没有随着维数的增加而衰减,始终可以找到相应函数的全局最优解. 对于函数 $f_8(x)$ 和 $f_9(x)$,4种算法在不同维数下均能找到全局最优解. 仿真结果表明,在10维、30维、50维和100维的测试函数上,UGBA均表现出了广泛的适应性,充分表明了改进后算法(UGBA)的收敛精度有了很大的提高.

表 8 4种算法对 $f_7(x)$ 的仿真结果

维数	算法	最差解	最优解	平均值
10	BA	4.542e-09	1.047e-09	2.367e-09
	DLBA	7.626e-14	3.849e-14	5.125e-14
	IBA	5.124e-15	1.982e-15	3.491e-15
	UGBA	**5.214e-15**	**3.159e-15**	**4.060e-15**
30	BA	3.426e-07	1.854e-07	2.521e-07
	DLBA	7.526e-13	1.249e-13	4.662e-13
	IBA	3.131e-14	2.302e-14	2.737e-14
	UGBA	**2.898e-14**	**1.775e-14**	**1.972e-14**
50	BA	9.364e-06	5.218e-06	6.656e-06
	DLBA	4.906e-11	3.368e-11	4.001e-11
	IBA	3.021e-12	1.512e-12	2.236e-12
	UGBA	**5.018e-13**	**8.569e-14**	**2.472e-13**
100	BA	4.309e-04	2.804e-04	3.560e-04
	DLBA	1.746e-09	4.142e-09	3.176e-09
	IBA	6.022e-10	3.791e-10	4.526e-10
	UGBA	**9.465e-11**	**5.641e-11**	**7.054e-11**

3.2 收敛曲线分析

算法的收敛曲线直观地反映了算法陷入局部极值的次数和向下收敛的速度,是衡量算法性能的重要指标. 限于篇幅,这里给出4种算法在维数 $n = 30$ 下的4个测试函数的收敛曲线,如图2～图5所示.

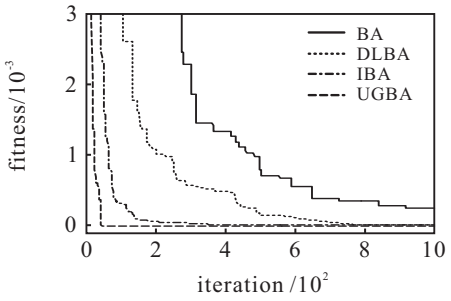

图 2 Griewank函数的收敛曲线

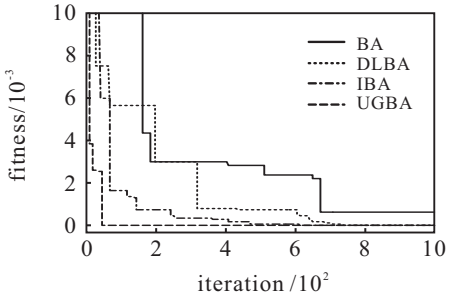

图 3 Ackley's函数的收敛曲线

由图2和图3可见,UGBA在50代之内就能够最终收敛,而其他3种算法在300～1 000代才能最终收敛. 对于函数 $f_8(x)$ 和 $f_9(x)$,4种算法均能找到全局最优解,但由图4和图5可见,UGBA的收敛速度远快

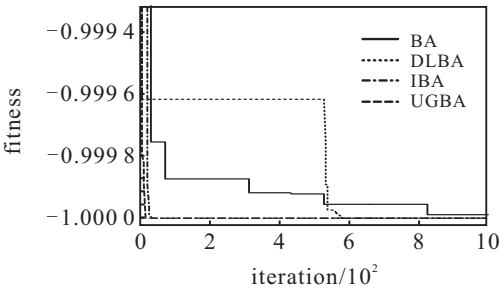

图 4 Easom's函数的收敛曲线

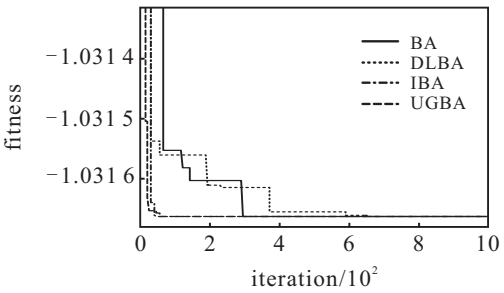

图 5 Six-Hump Camel-Back函数的收敛曲线

于BA和DLBA,UGBA和IBA的收敛曲线几乎重合,收敛速度稍快于IBA. 分析4种算法的收敛曲线发现,UGBA的收敛曲线形态接近垂直向下,出现弧形和阶梯形的次数明显少于其他3种算法. 这是因为算法根据变异开关函数在前期利用均匀变异能够快速收敛到最优解区域,在后期利用高斯变异跳出局部极值并在局部最优附近进行深度挖掘,快速收敛到全局最优解. 仿真结果表明,UGBA在处理多维多极值的问题上依然保持较高的寻优效率,表明了改进后算法(UGBA)的收敛速度比其他3种算法有显著的提高.

4 结 论

本文针对基本蝙蝠算法易陷入局部极值、寻优精度低、收敛速度慢的缺点,通过引入变异开关函数使蝙蝠个体在整个寻优过程中能够保证有一定的概率发生变异,保持了种群的活跃性. 算法中的均匀变异和高斯变异同时嵌入到整个寻优过程,较好地解决了基本蝙蝠算法前期定位最优解区域慢、后期易陷入局部极值、寻优精度低的问题. 仿真结果表明,UGBA的寻优精度和收敛速度均有较大提升.

参考文献(References)

[1] Eberhart R, Kennedy J. A new optimizer using particle swarm theory[C]. Proc of the 6th Int Symposium on Micro Machine and Human Science. Piscataway: IEEE, 1995: 39-43.

[2] Kennedy J, Eberhart R. Particle swarm optimization[C]. Proc of IEEE Int Conf on Neural Networks. Piscataway: IEEE, 1995: 1942-1948.

[3] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents[J]. IEEE

Trans on Systems, Man, and Cybernetics, Part B: Cybernetics a Publication of the IEEE Systems, Man, and Cybernetics Society, 1996, 26(1): 29-41.

[4] Goldberg D E. Genetic algorithm in search, optimization and machine learning[M]. Boston: Addison-Wesley Longman Publishing, 1989: 7-25.

[5] Pan W T. A new fruit fly optimization algorithm: Taking the financial distress model as an example[J]. Knowledge-Based Systems, 2012, 26(2): 69-74.

[6] Eusuff M M, Lansey K E. Optimization of water distribution network design using the shuffled frog leaping algorithm[J]. J of Water Sources Planning and Management, 2003, 129(3): 210-225.

[7] Yang X S. A new metaheuristic bat-inspired algorithm[C]. Nature inspired cooperative strategies for optimization (NISCO 2010), Studies in Computational Intelligence284. Berlin: Springer-Verlag, 2010: 65-74.

[8] Adarsh B R, Raghunathan T, Jayabarathi T, et al. Economic dispatch using chaotic bat algorithm[J]. Energy, 2016, 96(2): 666-675.

[9] Rahimi A, Bavafa F, Aghababaei S, et al. The online parameter identification of chaotic behaviour in permanent magnet synchronous motor by Self-Adaptive Learning Bat-inspired algorithm[J]. Int J of Electrical Power & Energy Systems, 2016, 78(6): 285-291.

[10] Karri Chiranjeevi, Jena U. Fast vector quantization using a Bat algorithm for image compression[J]. Engineering Science & Technology An International J, 2016, 19(2): 769-781.

[11] Sharma S P K M. Radially optimized zone-divided energy-aware wireless sensor networks(WSN) protocol using BA(bat algorithm)[J]. Iete J of Research, 2015, 61(2): 1-10.

[12] Wang G G, Chu H C E, Mirjalili S. Three-dimensional path planning for UCAV using an improved bat algorithm[J]. Aerospace Science & Technology, 2016, 49(2): 231-238.

[13] Luo J, Liu L, Wu X. A double-subpopulation variant of the bat algorithm[J]. Applied Mathematics & Computation, 2015, 263(7): 361-377.

[14] 陈梅雯, 钟一文, 王李进. 一种求解多维全局优化问题的改进蝙蝠算法[J]. 小型微型计算机系统, 2015, 36(12): 2749-2753.
(Chen M W, Zhong Y W, Wang L J. Improved bat algorithm for solving multi-dimensional global optimization problems[J]. J of Chinese Computer Systems, 2015, 36(12): 2749-2753.)

[15] He X, Ding W J, Yang X S. Bat algorithm based on simulated annealing and Gaussian perturbations[J]. Neural Computing & Applications, 2013, 25(2): 459-468.

[16] Wang X, Wang W, Wang Y. An adaptive bat algorithm[M]. Intelligent Computing Theories and Technology. Berlin: Springer Heidelberg, 2013: 216-223.

[17] 肖辉辉, 段艳明. 基于DE算法改进的蝙蝠算法的研究及应用[J]. 计算机仿真, 2014, 31(1): 272-277.
(Xiao H H, Duan Y M. Research and application of improve bat algorithm based on DE algorithm[J]. Computer Simulation, 2014, 31(1): 272-277.)

[18] Chawla M, Duhan M. Bat algorithm: A survey of the state-of-the-art[J]. Applied Artificial Intelligence, 2015, 29(6): 617-634.

[19] 鲁姝颖. 粒子群优化算法的几种改进算法及应用[D]. 徐州: 中国矿业大学理学院, 2014: 1-53.
(Lu S Y. Some improvements and applications of particle swarm optimization algorithm[D]. Xuzhou: College of Sciences, China University of Mining Technology, 2014: 1-53.)

[20] 郑云水, 岳小雪, 林俊亭. 带有高斯变异的混合蛙跳蝙蝠算法[J]. 计算机应用研究, 2015, 32(12): 3629-3633.
(Zheng Y S, Yue X X, Lin J T. Shuffled frog leaping and bat algorithm with Gauss mutation[J]. Application Research of Computers, 2015, 32(12): 3629-3633.)

[21] Yang X S, Gandomi A H. Bat algorithm: A novel approach for global engineering optimization[J]. Engineering Computations, 2012, 29(5): 464-483.

[22] Xie J, Zhou Y, Chen H. A novel bat algorithm based on differential operator and lévy flights trajectory[J]. Computational Intelligence & Neuroscience, 2013, 2013(3): 1-13.

[23] Yilmaz S, Kucuksille E U. Improved bat algorithm(IBA) on continuous optimization problems[J]. Lecture Notes on Software Engineering, 2013, 1(3): 279-283.

[24] 王皓, 欧阳海滨, 高立群. 一种改进的全局粒子群优化算法[J]. 控制与决策, 2016, 31(7): 1161-1168.
(Wang H, Ouyang H B, Gao L Q. An improved global particle swarm optimization[J]. Control and Decision, 2016, 31(7): 1161-1168.)

[25] Meng X B, Gao X Z, Liu Y, et al. A novel bat algorithm with habitat selection and Doppler effect in echoes for optimization[J]. Expert Systems with Applications, 2015, 42(17/18): 6350-6364.

（责任编辑：郑晓蕾）