一种动态调整惯性权重的自适应蝙蝠算法

裴宇航[1] 刘景森[2,3] 李 煜[4]

(河南大学计算机与信息工程学院 开封 475004)[1] (河南大学复杂智能网络系统研究所 开封 475004)[2]

(河南大学软件学院 开封 475004)[3] (河南大学管理科学与工程研究所 开封 475004)[4]

摘 要 为了加快蝙蝠算法的收敛速度并提高寻优精度,提出一种动态调整惯性权重的自适应蝙蝠算法。该算法在速度公式中加入惯性权重,并采用一种服从均匀分布和贝塔分布的随机调整策略,动态地调整惯性权重的大小,以加快算法的收敛速度。另外,引入了速度纠正因子,在每次迭代时,算法可根据当前种群的迭代次数动态地约束每一代蝙蝠的移动步长,从而使算法具有一定的自适应性。仿真实验结果表明,改进后的算法的寻优性能显著提高,具有较快的收敛速度和较高的寻优精度。

关键词 蝙蝠算法,惯性权重,速度纠正因子,自适应

中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.06.041

Adaptive Bat Algorithm with Dynamically Adjusting Inertia Weight

PEI Yu-hang[1] LIU Jing-sen[2,3] LI Yu[4]

(College of Computer and Information Engineering, Henan University, Kaifeng 475004, China)[1]

(Institute of Complex Intelligent Network System, Henan University, Kaifeng 475004, China)[2]

(School of Software, Henan University, Kaifeng 475004, China)[3]

(Research Institute of Management Science and Engineering, Henan University, Kaifeng 475004, China)[4]

Abstract In order to improve the performance and the precision of bat algorithm(BA), an adaptive bat algorithm with dynamically adjusting inertia weight(DAWBA) was proposed. Inertia weight which obeys the uniform distribution and beta distribution in the iteration formula is added into the algorithm, thus accelerating the convergence speed. In addition, we introduced a speed correction factor and used it to constraint the step of bat dynamically, which provides the algorithm with effective adaptability. Simulation results show that the performance of DAWBA is significantly improved.

Keywords Bat algorithm, Inertia weight, Speed correction factor, Adaptability

1 引言

近年来,随着人们对自然界生物群体的关注不断深入,受自然界中不同生物的启发,许多学者提出了群体智能算法,例如,受鸟群觅食行为启发提出的粒子群算法[1-2],受生物进化演变规律的启发提出的遗传算法[3],受蚁群在寻找食物过程中发现路径行为的启发提出的蚁群算法[4],受青蛙群体寻找食物移动规律的启发提出的混合蛙跳算法[5],受果蝇觅食行为的启发提出的果蝇优化算法[6]等。这些不同的算法在解决各类复杂优化问题上有着不俗的表现。

蝙蝠算法[7]是一种新型的元启发式全局优化算法,是由学者YANG X S于2010年提出的。蝙蝠算法是通过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,在该算法中,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TSP[8]、PMSM(永磁同步电机)参数识别[9]、图像识别[10]、路径规划[11]、工艺过程设计[12]等问题。

蝙蝠算法有模型简单、算法参数少、通用性强等优势,但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等缺点。针对这些不足,国内外许多学者对蝙蝠算法做出了相应的改进,如文献[13]把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,从而提高了种群的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]利用立方映射产生混沌序列,对蝙蝠的位置和速度进行初始化,提高了种群的全局搜索能力。文献[15]把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

到稿日期:2016-05-10 返修日期:2016-07-22 本文受河南省科技厅科技攻关项目(162102110109),河南省科技攻关重点项目(142102210036)资助。

裴宇航(1990—),男,硕士,主要研究方向为智能算法,E-mail:peiyuhang258@qq.com;刘景森(1968—),男,博士,教授,主要研究方向为计算机网络、智能算法、电子商务,E-mail:ljs@henu.edu.cn(通信作者);李 煜(1969—),女,博士,教授,主要研究方向为智能算法、电子商务。

好的全局搜索能力而且还加快了算法的收敛速度。文献[16]在速度公式第一项设置自适应惯性权重，使蝙蝠在搜索过程中能动态地调整速度。文献[17]把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，本文提出一种动态调整惯性权重的自适应蝙蝠算法(DAWBA)，改进后的算法可以动态地赋予蝙蝠个体随机的速度值，这既加快了种群定位到最优解区域的速度，又提高了种群跳出局部极值的。仿真表明，改进后的算法具有较快的收敛速度和较高的寻优精度。

2　基本蝙蝠算法

小型蝙蝠视力退化，基本上靠声纳来辨认物体。人们通过长时间观察研究蝙蝠捕食猎物以及其飞行特征，发现蝙蝠是通过改变发声的响度和频率来辨别物体的形态[18]，根据返回的超声波的变化来判断物体的距离、移动方向、速度大小等。根据蝙蝠的回声定位特性，Yang 教授提出了蝙蝠算法。蝙蝠算法就是模拟蝙蝠捕食，最终找到最优解的过程。算法给出由 n 只蝙蝠组成的群体，每只蝙蝠的频率、速度和位置更新公式如下：

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

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

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

其中，f_i 表示第 i 只蝙蝠发出的频率，$f_{\min}$ 和 $f_{\max}$ 表示蝙蝠发出频率的最小值和最大值。β 是在[0,1]之间均匀分布的随机向量。$f_i\in[f_{\min},f_{\max}]$，$x_i$ 和 v_i 分别表示在搜索空间中第 i 只蝙蝠的位置和速度，$i=1,2,3,\cdots,n$。x_* 表示当前全局的最优解。

在局部搜索部分，局部位置更新公式为：

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

其中，ε 是在[−1,1]之间的一个随机数。A^t 是这一代所有蝙蝠响度的平均值。

蝙蝠发出的脉冲响度 A^t 和脉冲发射速率 r_i 随着迭代进行更新，一旦蝙蝠发现猎物，响度就会逐渐降低，同时提高脉冲发出的速率。响度可以用任意合适的值表示。响度 A_i^t 和发射速率 r_i 的更新公式为：

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

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

其中，α 和 γ 是常数，$0<\alpha<1$，$\gamma>0$。

3　动态调整惯性权重的自适应蝙蝠算法

3.1　构造随机惯性权重

从基本蝙蝠算法中的式(2)可以看出，速度项系数恒定为1，这样大大降低了蝙蝠的灵活性以及蝙蝠种群的多样性，并且导致了算法的全局搜索和局部搜索不均衡。为了解决该问题，王文等人[19]提出了一种动态改变惯性权重的策略，即非线性地减小 ω 的取值，惯性权重表示为：

$$\omega(t)=\omega_{\min}+(\omega_{\max}-\omega_{\min})\cdot\exp(-\rho(\frac{t}{T_{\max}})^2) \tag{7}$$

其中，$\omega_{\max}$ 和 $\omega_{\min}$ 为 $\omega(t)$ 的最大值和最小值，$1<\rho<T_{\max}$，$T_{\max}$ 为最大迭代次数。$\omega(t)$ 使蝙蝠前期的搜索经验为后期的搜索提供支持，从而加快了算法的收敛速度。但是，从式(7)可以明显看出，虽然递减的惯性权重赋予算法前期较快的速度，后期较慢的速度，但是却忽略了每一代内所有蝙蝠的速度依旧恒定不变的问题，即种群的每一代都缺乏多样性。这些不足会造成算法的迭代次数增加，收敛速度变慢。为了解决这个问题，受到 CHEN Z 等人[20]的启发，本文提出一种随机的惯性权重函数，具体表达式为：

$$\omega=\omega_{\max}-(\omega_{\max}-\omega_{\min})*rand()-\tau*betarnd() \tag{8}$$

其中，$\omega_{\max}$ 和 $\omega_{\min}$ 为 ω 的最大值和最小值，τ 为惯性偏离因子，$\tau\in[0.1,0.9]$，$rand()$ 为[0,1]上的均匀分布，$betarnd()$ 为服从贝塔分布产生的随机数[21]，其分布覆盖率包含了从均匀分布到正态分布即各种不对称的分布，即取值在(0,1)之间的几率模式。惯性权重 ω 的表达式中的第二项通过 $rand()$ 以相同的概率赋予($\omega_{\max}-\omega_{\min}$)系数，从而使 ω 随机地在[$\omega_{\min}$，$\omega_{\max}$]上取值，最后一项利用贝塔分布使 ω 偏离，从而使惯性权重的取值分布更均匀，也更灵活。在 $betarnd()$ 前加入 τ(惯性偏离因子)是为了控制 ω 的偏离程度，使偏离更合理。总的来说，这样的惯性权重不仅在代与代之间具有随机性，而且每一代内的蝙蝠也具有随机性，增强了每一代内蝙蝠种群的多样性，一定程度上缓解了蝙蝠算法收敛速度慢、易陷入局部极值等问题。速度更新公式变为：

$$v_i^t=\omega*v_i^{t-1}+(x_i^{t-1}-x_*)f_i \tag{9}$$

3.2　构造速度纠正因子

本文赋予惯性权重随机的数值，从而使每一代蝙蝠的速度具有很强的随机性，这样可能会导致蝙蝠每一代的移动都太过“随意”，为了使算法在进化过程中更加稳定，提出了速度纠正因子函数，表达式为：

$$c_i(t)=1-\frac{t}{T_{\max}+rand()} \tag{10}$$

位置更新公式变为：

$$x_i^t=x_i^{t-1}+c_i(t)*v_i^t \tag{11}$$

其中，t 为算法的迭代次数，$T_{\max}$ 为算法设定的最大迭代次数，$c_i(t)\in(0,1)$。$c_i(t)$ 随着迭代次数的增加而逐渐减小，这有利于前期全局搜索以及后期局部深度挖掘。加入 $c_i(t)$ 后，根据迭代次数动态地约束每一代太过“随意”移动的蝙蝠，使每只蝙蝠的移动速度不仅具有随机性，而且具有稳定性，也使算法具有了一定的自适应性。

3.3　算法流程

Step 1　随机初始化蝙蝠种群个体数 n，蝙蝠的位置 x_i，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　根据适应度函数 $f(x)$，$x=(x_1,x_2,x_3,\cdots,x_n)^{\mathrm{T}}$，求出每只蝙蝠的适应度函数值，并找出最优的值。记录蝙蝠最优值的位置为 x_*。

Step 3　根据式(1)、式(8)、式(9)、式(11)更新蝙蝠的位置和速度。

Step 4　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。

Step 5　产生一个随机数 $rand$，如果 $rand<A^t$ 且 $f(x_i)<f(x_*)$，那么就将 Step 4 产生的新解记为当前最优解。然后

按照式(5)和式(6)更新响度 A_i^t 和脉冲发射速率 r_i。

Step 6 把所有蝙蝠的适应度值排序，找出当前的最优解 x_*。

Step 7 重复 Step2－Step6 的迭代过程，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 8 输出全局最优解。

4 实验及结果分析

为了全面测试本文提出的动态调整惯性权重的自适应蝙蝠算法(DAWBA)的寻优性能，本文选取了 10 个常用的经典测试函数[22-24]与基本蝙蝠算法(BA)[25]和非线性递减惯性权重的蝙蝠算法(Bat Algorithm With Nonlinear Decreasing Inertia Weight，NLDWBA)[19]进行对比测试。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,…,0)处取得最小值 0。

(2)Ronsenbrock 函数

$$f_2(x)=\sum_{i=1}^{n}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$$

该函数在(1,…,1)处取得最优值 0。

(3)Griewank 函数

$$f_3(x)=1+\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$$

该函数在(0,0)处取得最小值 0。

(4)Hansen 函数

$$f_4(x)=\sum_{i=1}^{n}i\cdot\cos((i-1)\cdot x_1+i)\cdot\sum_{j=1}^{n}j\cdot\cos((j+1)\cdot x_2+j)$$

该函数在(－7.589893，－7.708314)，(－7.589893，－1.425128)，(－7.589893，4.858057)，(－1.306708，－7.708314)，(－1.306708，－1.425128)，(－1.306708，4.858057)，(4.976478，－7.708314)，(4.976478，4.858057)，(4.976478，－1.425128)处取得最小值－176.541793。

(5)Schwefel's problem 2.22 函数

$$f_5(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|$$

该函数在(0,…,0)处取得最小值 0。

(6)Ackley's 函数

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

该函数在(0,…,0)处取得最小值 0。

(7)Branin 函数

$$f_7(x)=\left(x_2-\frac{5.1}{4\pi}x_1^2-6\right)^2+10\cdot\left(1-\frac{1}{8\pi}\right)\cos x_1+10$$

该函数在(π，12.275)，(π，2.275)，(9.42478，2.475)处取得最小值 0.39789。

(8)Rastrigin 函数

$$f_8(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$$

该函数在(0,…,0)取得最优值 0。

(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(－0.0898，0.7126)和(0.0898，－0.7126)处取得最优值－1.0316285。

(10)Goldstein-Price 函数

$$f_{10}(x)=[1+(x_1+x_2+1)^2\cdot(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2^2)^2\cdot(18-32x_1+12x_1^2+48x_2^2-36x_1x_2+27x_2^2)]$$

该函数在(0，－1)取得最小值 3。

4.1 寻优精度分析

为了保证客观性和公正性，3 种算法的参数设置基本一致，具体如下：种群规模 $n=40$，响度最大值 A_0 取 0.25，脉冲最大值 r_0 取 0.5，最大迭代次数为 1000。对于每个测试函数，每种算法独立运行 30 次。为了进一步证明本文所提算法的优越性，统计了 3 种算法运行 10 个测试函数的最差解、最优解和平均值，并加粗了精度最高的解，具体实验结果如表 1 所列。

表 1 寻优精度的对比分析

函数	算法	最差解	最优解	平均值
$f_1(x)$	BA	3.822e－08	1.350e－08	2.956e－08
	NLDWBA	8.930e－10	4.050e－10	6.273e－10
	DAWBA	**2.791e－10**	**6.224e－11**	**4.628e－11**
$f_2(x)$	BA	0.004881	0.002792	0.003350
	NLDWBA	2.644e－06	1.492e－06	2.501e－06
	DAWBA	**8.757e－08**	**7.089e－08**	**9.762e－08**
$f_3(x)$	BA	4.167e－09	3.106e－09	3.441e－09
	NLDWBA	7.707e－10	1.185e－10	2.557e－10
	DAWBA	**2.564e－11**	**1.307e－11**	**2.185e－11**
$f_4(x)$	BA	－176.5418	－176.5418	－176.5418
	NLDWBA	－176.5418	－176.5418	－176.5418
	DAWBA	－176.5418	－176.5418	－176.5418
$f_5(x)$	BA	0.000416	0.000201	0.000263
	NLDWBA	5.622e－05	2.753e－05	3.046e－05
	DAWBA	**1.837e－05**	**1.680e－05**	**1.751e－05**
$f_6(x)$	BA	0.000377	0.000128	0.000301
	NLDWBA	5.193e－05	4.474e－05	6.735e－05
	DAWBA	**3.329e－05**	**1.544e－05**	**2.803e－05**
$f_7(x)$	BA	0.39789	0.39789	0.39789
	NLDWBA	0.39789	0.39789	0.39789
	DAWBA	0.39789	0.39789	0.39789
$f_8(x)$	BA	5.039e－12	8.046e－13	2.824e－12
	NLDWBA	2.993e－12	1.401e－12	1.914e－12
	DAWBA	**1.296e－13**	**7.105e－15**	**1.954e－14**
$f_9(x)$	BA	－1.0316	－1.0316	－1.0316
	NLDWBA	－1.0316	－1.0316	－1.0316
	DAWBA	－1.0316	－1.0316	－1.0316
$f_{10}(x)$	BA	3	3	3
	NLDWBA	3	3	3
	DAWBA	3	3	3

由表 1 可以看出，对于测试函数 $f_4(x)$，$f_7(x)$，$f_9(x)$，$f_{10}(x)$，3 种算法均能够找到最优解，寻优精度无差别。但对于其余测试函数，本文提出的动态调整惯性权重的自适应蝙蝠算法(DAWBA)在最差解、最优解及平均值的精度方面均明显高于基本蝙蝠算法(BA)和非线性递减惯性权重的蝙蝠算法(NLDWBA)，从而证明了本文算法的寻优精度比其他两种算法有一定程度的提高。

4.2 收敛速度分析

算法的收敛速度直接反映了算法跳出局部极值的能力，是衡量算法性能的重要指标，图 1－图 10 为 3 种算法寻优过程的对比仿真图。

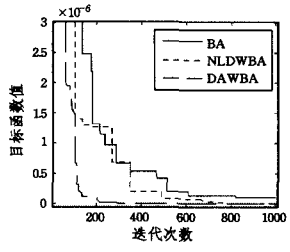

图 1　Sphere 函数的收敛曲线

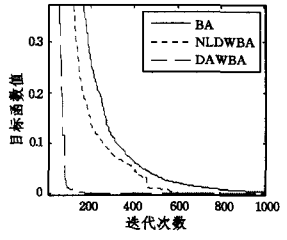

图 2　Ronsenbrock 函数的收敛曲线

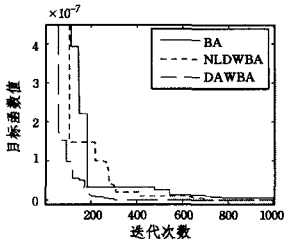

图 3　Griewank 函数的收敛曲线

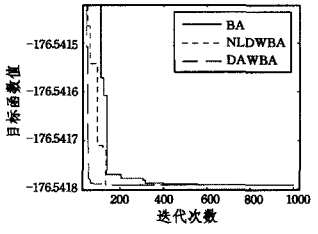

图 4　Hansen 函数的收敛曲线

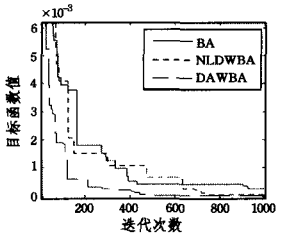

图 5　Schwegel's problem2. 22 函数的收敛曲线

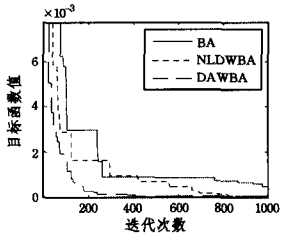

图 6　Ackley's 函数的收敛曲线

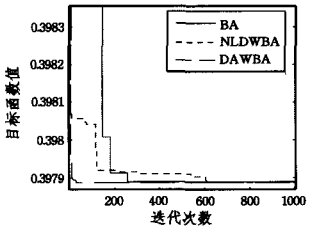

图 7　Branin 函数的收敛曲线

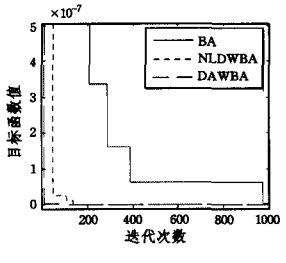

图 8　Rastrigin 函数的收敛曲线

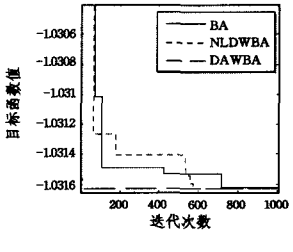

图 9　Six-Hump Camel-Back 函数的收敛曲线

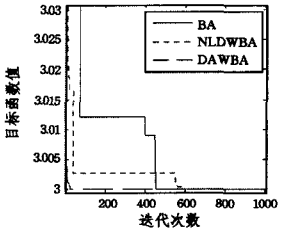

图 10　Goldstein-Price 函数的收敛曲线

$f_1(x)$和 $f_2(x)$为单峰函数，如图 1、图 2 所示，本文算法(DAWBA)能够在 100～300 代内最终收敛，而其他两种算法在 600～1000 代才能最终收敛。$f_2(x)$－$f_{10}(x)$为多峰函数，$f_3(x)$，$f_5(x)$和 $f_6(x)$为地形复杂的多峰函数，存在许多局部极小值，容易早熟收敛，从图 3、图 5 和图 6 可以看出 3 种算法均多次陷入局部极值，但本文算法均能够在短时间内跳出局部极值，快速向下收敛。对于函数 $f_4(x)$，$f_7(x)$，$f_9(x)$，$f_{10}(x)$，虽然 3 种算法均能顺利地找到全局最优解，但分析它们的寻优曲线(图 4、图 7、图 9、图 10)发现，本文算法陷入局部极值的次数均明显少于其他两种算法，且本文算法在40～100 代就能够最终收敛，远快于其他两种算法。综上所述，本文算法均能很快趋于稳定，很少出现震荡。这是因为随机的惯性权重使每一代内的蝙蝠速度不同，降低了陷入局部极值的概率，从而能够快速收敛。

结束语　本文针对蝙蝠算法易陷入局部最优值、收敛速度慢等缺点，提出了一种动态调整惯性权重的自适应蝙蝠算法(DAWBA)。该算法赋予每一只蝙蝠随机的速度，同时引入速度纠正因子来约束蝙蝠的移动步长，这样既提高了蝙蝠种群的多样性，又使得算法具有自适应性。仿真结果表明，本文算法(DAWBA)具有较快的收敛速度，寻优精度也有较大的提高。

参 考 文 献

[1] EBERHART R, KENNEDY J. A new optimizer using particle swarm theory[C]// Proceedings of the Sixth International Symposium on Micro Machine and Human Science. 1995: 39-43.

[2] KENNEDY J, EBERHART R. Particle swarm optimization [C] // Proceedings of IEEE International Conference on Neural Networks. 1995: 1942-1948.

[3] GOLDBERG D E. Genetic algorithm in search, optimization and machine learning [M]. Boston: Addison-Wesley Longman Publishing Co. Inc, 1989.

[4] DORIGO M, MANIEZZO V, COLORNI A. Ant system: optimization by a colony of cooperating agents[C] // IEEE Transactions on Systems Man & Cybernetics. 1996: 29-41

[5] EUSUFF M M, LANSEY K E. Optimization of Water Distribution Network Design Using the Shuffled Frog leaping Algorithm [J]. Journal of Water Sources Planning and Management, 2003, 129(3): 210-225.

[6] 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.

[7] YANG X S. A New meta heuristic Bat-Inspired Algorithm[M]// Nature Inspired Cooperative Strategies for Optimization (NISCO 2010). Berlin Eidelberg: Springer-Verlag, 2010: 65-74.

[8] OOSABA E, YANG X S, DIAZ F, et al. An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems [J]. Engineering Applications of Artificial Intelligence, 2016, 48: 59-71.

[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]. International Journal of Electrical Power & Energy Systems, 2016, 78(c): 285-291.

[10] GAO M L, SHEN J, YIN L J, et al. A Novel Visual Tracking

Method Using Bat Algorithm [J]. Neurocomputing, 2015, 177 (c): 612-619.

[11] 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: 231-238.

[12] WANG J, FAN X, ZHAO A, et al. A Hybrid Bat Algorithm for Process Planning Problem[J]. IFAC-Papersonline, 2015, 48(3): 1708-1713.

[13] LUO J, LIU L, WU X. A double-subpopulation variant of the bat algorithm [J]. Applied Mathematics & Computation, 2015, 263(C): 361-377.

[14] YIN J T, LIU Y L, LIU L, et al. Efficient hybrid bat algorithm [J]. Computer Engineering and Applications, 2014, 50(7): 62-66. (in Chinese)
尹进田，刘云连，刘丽，等. 一种高效的混合蝙蝠算法[J]. 计算机工程与应用，2014，50(7)：62-66.

[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. Springer Berlin Heidelberg, 2013: 216-223.

[17] 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. (in Chinese)
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.

[18] LIU C P, YE C M, LIU M C. Optimization strategy from nature: perceive as bat [J]. Application Research of Computers, 2013, 30(5): 1320-1322, 1356. (in Chinese)
刘长平，叶春明，刘满成. 来自大自然的寻优策略：像蝙蝠一样感知[J]. 计算机应用研究，2013，30(5)：1320-1322，1356.

[19] WANG W, WANG Y, WANG X W. An Improved Bat Algorithm with Memory Characteristic [J]. Computer Application and Software, 2014, 31(11): 257-259, 329. (in Chinese)
王文，王勇，王晓伟. 一种具有记忆特征的改进蝙蝠算法[J]. 计算机应用与软件，2014，31(11)：257-259，329.

[20] CHEN Z, YONG Q Z, LU M D. A Simplified-Adaptive Bat Algorithm Based on Frequency [J]. Journal of Computational Information Systems, 2013, 9(16): 6451-6458.

[21] PANT M, THANGARAJ R, ABRAHAM A. Particle swarm optimization using adaptive mutation [C]// Proc of 19th International Workshop on Database and Expert Systems Application. Turin: IEEE, 2008: 519-523.

[22] LI Y, MA L. Bat-inspired Algorithm: A Novel Approach for Global Optimization [J]. Computer Science, 2013, 40(9): 225-229. (in Chinese)
李煜，马良. 新型全局优化蝙蝠算法[J]. 计算机科学，2013，40(9)：225-229.

[23] JORDEHI A R. Chaotic bat swarm optimization (CBSO)[J]. Applied Soft Computing, 2014, 26(c): 523-530.

[24] 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.

[25] YANG X S. Bat algorithm for multi-objective optimization [J]. International Journal of Bio-Inspired Computation, 2011, 3(5): 267-274.

(上接第 215 页)

[4] HSIEH C J, CHANG K W, LIN C J, et al. A dual coordinate descent method for large-scale linear SVM[C]// International Conference on Machine Learning. Helsinki, Finland: IEEE press, 2008: 1369-1398.

[5] CRAMER J S. The origins of logistic regression: 02-119/4[R]. Uinkeveren: Tinbergen Institute, 2002.

[6] PLATT J. Sequential minimal optimization: A fast algorithm for training support vector marchines [J]. Journal of Information Technology, 1998, 2(5): 1-28.

[7] BOYDSL, VANDENBERGHE. Convex Optimization[M]. Cambridge, UK: Cambridge University Press, 2004.

[8] DIETTERICH T G. Machine learning research: Four current directions [J]. AI Magazine, 1997, 18(4): 97-136.

[9] ZHOU Z H, WU J X, TANG W. Ensembling neural networks: Many could be better than all[J]. Artificial Intelligence, 2002, 13 (1/2): 239-263.

[10] ZHANG C X, ZHANG J S. A Survey of Selective Ensemble Learning Algorithms [J]. Chinese Journal of Computer, 2011, 34 (8): 1399-1410. (in Chinese)
张春霞，张讲社. 选择性集成学习算法综述[J]. 计算机学报，2011，34(8)：1399-1410.

[11] FREUND Y, ROBERT E S. A decision-theoretic generalization of on-line learning and an application to boosting [J]. Journal of Computer and System Sciences, 1997, 55(1): 119-139.

[12] BREIMAN L. Bagging predictors [J]. Machine Learning, 1996, 24(2): 123-140.

[13] BREIMAN L. Random forests [J]. Machine Learning, 2001, 45 (1): 5-32.

[14] ZHOU Z H. Machine Learning[M]. Beijing: Tsinghua University Press, 2016. (in Chinese)
周志华. 机器学习[M]. 北京：清华大学出版社，2016.

[15] LI H. Statistical Learning Method [M]. Beijing: Tsinghua University Press, 2012. (in Chinese)
李航. 统计学习方法[M]. 北京：清华大学出版社，2012.

[16] LI Y, SI J, ZHOU G J, et al. FREL: A Stable Feature Selection Algorithm [J]. IEEE Trains. Neural Netw, 2015, 26(7): 1388-1402.

[17] LU H J, AN C L. Disagreement Measure Based Ensemble of Extreme Learning Machine for Gene Expression Data Classification [J]. Chinese Journal of Computer, 2013, 36(2): 341-348. (in Chinese)
陆慧娟，安春霖. 基于输出不一致测度的极限学习机集成的基因表达数据分类[J]. 计算机学报，2013，36(2)：341-348.