基于模拟退火高斯扰动的蝙蝠优化算法*

贺兴时[1]，丁文静[1]，杨新社[1,2]
(1. 西安工程大学 理学院，西安 710048；2. 密德萨斯大学 科学与技术学院，伦敦 NW4 4BT)

摘　要：蝙蝠算法(bat algorithm，BA)是一类新型的搜索全局最优解的随机优化技术。为了提高 BA 算法的搜索效果，把模拟退火的思想引入到蝙蝠优化算法中，并对蝙蝠算法的某些个体进行高斯扰动，提出了一种基于模拟退火的高斯扰动蝙蝠优化算法(SAGBA)。分别将蝙蝠优化算法、模拟退火粒子群算法、SAGBA 在 20 个典型的基准测试函数中进行仿真对比，结果表明 SAGBA 不仅增加了全局收敛性，而且在收敛速度和精度方面均优于其他两种算法。

关键词：蝙蝠算法；模拟退火；高斯扰动；仿真；优化
中图分类号：TP18；TP301.6　　**文献标志码**：A　　**文章编号**：1001-3695(2014)02-0392-06
doi:10.3969/j.issn.1001-3695.2014.02.017

Bat algorithm based on simulated annealing and Gaussian perturbations

HE Xing-shi[1]，DING Wen-jing[1]，YANG Xin-she[1,2]
(1. *School of Science*，*Xi'an Polytechnic University*，*Xi'an* 710048，*China*；2. *School of Science & Technology*，*Middlesex University*，*London NW4 4BT*，*UK*)

Abstract：Bat algorithm is a new stochastic optimization technique for global optimization. This paper introduced both simulated annealing and Gaussian perturbations into the standard bat algorithm so as to enhance its search performance. As a result，it proposed a simulated annealing gaussian bat algorithm (SAGBA) for global optimization. It used BA，SAPSO and SAGBA to carry out numerical experiments for 20 test benchmarks. The simulation results show that the proposed SAGBA can indeed improve the global convergence. In addition，SAGBA is superior to the other two algorithms in terms of convergence and accuracy.

Key words：bat algorithm(BA)；simulate annealing(SA)；Gaussian perturbations；simulation；optimization

蝙蝠算法(BA)是 Yang 教授[1]于 2010 年基于群体智能提出的启发式搜索算法，是一种搜索全局最优解的有效方法。该算法是一种基于迭代的优化技术，初始化为一组随机解，然后通过迭代搜寻最优解，且在最优解周围通过随机飞行产生局部新解，加强了局部搜索。与其他算法相比，BA 在准确性和有效性方面远优于其他算法，且没有许多参数要进行调整。目前 BA 算法已被用于工程设计[2~4]、分类[5]、模糊聚类[6]、预测[7]和神经网络[8]等领域。本文提出了一种基于模拟退火的高斯扰动蝙蝠优化算法(SAGBA)，该算法既保持了蝙蝠优化算法简单易行且有效的特点，又增强了蝙蝠算法跳出局部最优解的能力，提高了算法的收敛速度。

1　蝙蝠算法

蝙蝠是唯一长着翅膀的哺乳动物，而且它们有先进的回音定位能力[9]。大多数微型蝙蝠是食虫类的动物。微型蝙蝠使用声波回声定位，检测到猎物，躲避障碍，并在黑暗中找到自己位于裂缝的栖息地。这些蝙蝠发出很响亮的声音，然后听到从周围物体反射回来的回音。

对于不同的蝙蝠，它们的脉冲是与狩猎的策略有关的。大多数蝙蝠通过一种滤波器用短且高频的信号扫描周围，而其他蝙蝠则经常使用固定频率的信号进行回声定位。其信号频宽的变化取决于蝙蝠的种类，且常常通过使用更多的谐波而增加。

使用理想化微型蝙蝠的一些回声定位的特点，蝙蝠理想化的规则如下：

a)所有蝙蝠利用回声定位的感觉差异来判断食物、猎物和障碍物之间的差异。

b)蝙蝠是以速度 v_i、位置 x_i 和固定频率 $f_{\min}$(或波长)随机飞行的，并用不同的波长(或频率 f)和音量 A_0 来搜索猎物。它们会根据猎物的接近程度自动调整其发出的脉冲波长(或频率)。

c)虽然音量在不同形式下变化不同，在这里假设音量从一个很大的(正数)A_0 变化到最小值 $A_{\min}$。

基于这些理想化的规则，蝙蝠算法的基本步骤可以概括如下：

a)初始化蝙蝠种群 x_i 和 v_i($i=1,2\cdots n$)
b)初始化频率 f_i、脉冲速率 r_i 和音量 A_i
c)while ($t<$最大迭代次数)
d)通过调整频率产生新的解，同时根据式(1)~(3)更新速度和位置
e)if ($\mathrm{rand}>r_i$)
f)从这些最优解集中选择一个解

收稿日期：2013-05-14；**修回日期**：2013-06-27　　**基金项目**：陕西省软科学基金资助项目(2012KRM58)；陕西省教育厅自然科学基金资助项目(12JK0744，11JK0188)；西安工程大学研究生创新基金资助项目(chx131115)

作者简介：贺兴时(1960-)，男，陕西富平人，教授，硕士，主要研究方向为智能优化算法、数理统计、数据挖掘等(xsh1002@126.com)；丁文静(1988-)，女，硕士研究生，主要研究方向为智能优化算法；杨新社(1965-)，男，教授，博士，主要研究方向为智能优化算法、数学建模.

g)在选择的这个最优解附近产生一个局部解

h)end if

i)通过随机飞行产生一个新解

j)if (rand $< A_i$ & $f(x_i) < f(x_*)$)

k)接受新解

l)增大 r_i 的值,减小 A_i 的值

m)end if

n)对蝙蝠进行排序,找到当前最优解 x_*

o)end while

蝙蝠算法一个明显的简化是用无限追踪来估计时间的延迟和三维地形,它在几何计算中的应用非常好,由于面临的应用大多是多维问题,因此这一特性在实际中很少用到。除了简化的假设,还可以使用一些近似值。一般情况下,频率 f 的范围是$[f_{\min}, f_{\max}]$,对应波长的范围是$[\lambda_{\min}, \lambda_{\max}]$。例如,一个频率的范围是 [20 kHz, 500 kHz],对应波长的范围就是 0.7 ~ 17 mm。

1.1　虚拟蝙蝠的运动

在模拟仿真中必须使用虚拟蝙蝠。在一个 d 维的搜索空间中,t 时刻,BA 中的第 i 只蝙蝠的位置 x_i^t 和速度 v_i^t 按以下公式进行更新[1]:

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

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

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

其中:f_i 为频率;$\beta \in [0,1]$是一个服从均匀分布的随机变量;x_* 为当前全局最优位置,它是搜索完所有蝙蝠得到的位置后比较得到的最优位置。由于乘积 $\lambda_i f_i$ 是速度增量,可以根据具体问题的需要,固定一个因素 λ_i(或 f_i),同时使用另一个因素 f_i(或 λ_i)来调整速度的改变。

对于局部搜索,一旦在当前最优解中选中了一个解,那么每只蝙蝠按照随机游走产生的局部新解为

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

其中:$\varepsilon \in [-1,1]$是一个随机数,$A^t = A_i^t$ 是所有蝙蝠在同一个时间段的平均音量。蝙蝠的速度和位置更新步骤有些类似于标准粒子群优化[10]。在一定程度上,BA 可以看做是标准粒子群优化与强化的局部搜索的平衡结合,其平衡是受音量和脉冲发生率的控制。

1.2　音量和脉冲发生率

音量 A_i 和脉冲发生率 r_i 按照以下迭代过程更新:当蝙蝠找到猎物时,音量就会降低,同时脉冲发生率就会增加,音量会以任意简便的值改变。例如,为简单起见,可用 $A_0 = 1, A_{\min} = 0$。假设 $A_{\min} = 0$ 意味着一个蝙蝠刚刚找到猎物并暂时停止发出任何声音。更新式为

$$A_i^{t+1} = \alpha A_i^t,\ r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{5}$$

其中:α 和 γ 为常量。实际上,α 类似于本文模拟退火中一个冷却过程的冷却因素。对任意的 $0 < \alpha < 1, \gamma > 0$,有

$$A_i^t \to 0,\ r_i^t \to r_i^0,\ \text{当}\ t \to \infty \tag{6}$$

最简单的情况是令 $\alpha = \gamma$,在实现中,本文使用的是 $\alpha = \gamma = 0.9 \sim 0.975$。

2　模拟退火

模拟退火(SA)算法是最简单、最流行的启发式算法之一,它是一个以轨迹为基础的全局随机搜索方法[11]。SA 算法是模拟材料加工中的退火过程,现已广泛应用于组合优化问题中。SA 算法在搜索过程中具有概率突跳的能力,能有效地避免在搜索过程中陷入局部极值,它在退火过程中不但接收好的解,而且还以一定的概率接收差的解,同时这种概率受到温度参数的控制,其大小随着温度的下降而变小。事实上,在足够缓慢的冷却速度下,SA 就会收敛于全局最优解。从本质来看,模拟退火属于一个在适当条件下收敛的 Markov 链的搜索算法,其伪码如下:

a)初始化温度 T_0 和初始解 x_0

b)设置最终温度 T_f 和最大迭代次数 N

c)定义冷却表 $T \mapsto \alpha T (0 < \alpha < 1)$

d)while ($T > T_f$, $n < N$)

e)随机产生新解 $x_{n+1} = x_n + \varepsilon$

f)计算 $\Delta f = f_{n+1}(x_{n+1}) - f_n(x_n)$

g)当新解较优时接收新解

h)if 不接收新解

i)产生一个随机数 r

j)if ($p = \exp[-\Delta f/T] > r$) 接受

k)end if

l)更新最优解 x^* 和最优值 f^*

m)$n = n + 1$;

n)end while

虽然目前已有很多研究者把模拟退火算法与其他优化算法结合起来[12~14],但尚未见把模拟退火算法与蝙蝠优化算法相结合的算法。

3　基于模拟退火的蝙蝠优化算法

将模拟退火思想引入蝙蝠优化算法中,提出一种基于模拟退火的高斯扰动蝙蝠优化算法(SAGBA),采用高斯扰动的变异运算进一步调整所优化的集群。其基本执行过程是:先随机产生初始种群,开始随机搜索,设置初始温度,利用模拟退火算法得到全局最优的替代值,再根据位置和速度更新式产生一组新解,然后在个体较优位置进行高斯扰动,进行进一步的搜索。

在每次进化中,变异运算根据高斯扰动进行变异,用高斯扰动后的位置取代原位置,即

$$\boldsymbol{X}^t = \boldsymbol{X}^t + a \oplus \varepsilon \tag{7}$$

其中:ε 和 $\boldsymbol{X}^t$ 是同阶的随机矩阵,每个元素 $\varepsilon_{ij} \sim N(0,1)$;$\oplus$即点乘。为了避免波动过大,$a$ 用来调节 ε 的搜索范围。

整个算法可以分成三个部分:a)用模拟退火算法找到全局最优的替代值;b)利用蝙蝠优化算法找到一组新解;c)对新解进行高斯扰动,进行进一步搜索。其算法步骤如下:

a)随机初始化种群的位置、速度、频率、脉冲速率和音量。

b)评价每只蝙蝠个体的适应度,存储当前各个体的位置和适应值,并将所有个体中适应值最优的位置和适应值存储于 pbest 中。

c)确定初始温度。

d)根据式(8)确定当前温度下各个体的适配值。

$$TF(x_i) = \frac{e^{-(f(x_i) - f(\text{pbest}))/t}}{\sum_{i=1}^{N} e^{-(f(x_i) - f(\text{pbest}))/t}} \tag{8}$$

e)采用轮盘赌策略从所有的蝙蝠个体中确定全局最优的

某个替代值 pbest′,然后根据式(9)更新各个体的速度和位置。

$$v_i^t = v_i^{t-1} + (x_i^t - \text{pbest}')f_i \tag{9}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{10}$$

f)计算各蝙蝠个体新的目标值,更新各个体的较优位置,在以上较优位置处进行高斯扰动,再比较高斯扰动前后的位置,找出最优位置 pbest 和对应的最优值。

g)进行退温操作。

h)若满足停止条件(通常为精度或迭代次数),搜索停止,输出结果,否则转到 d)。

i)初始温度和退温的方式对算法有一定的影响,一般采用如下初温和退火方式[15]:

$$t_{k+1} = \lambda t_k \tag{11}$$

$$t_0 = f(\text{pbest})/\ln 5 \tag{12}$$

其中:λ 为[0.5,1]之间的冷却系数。

算法流程如图 1 所示。

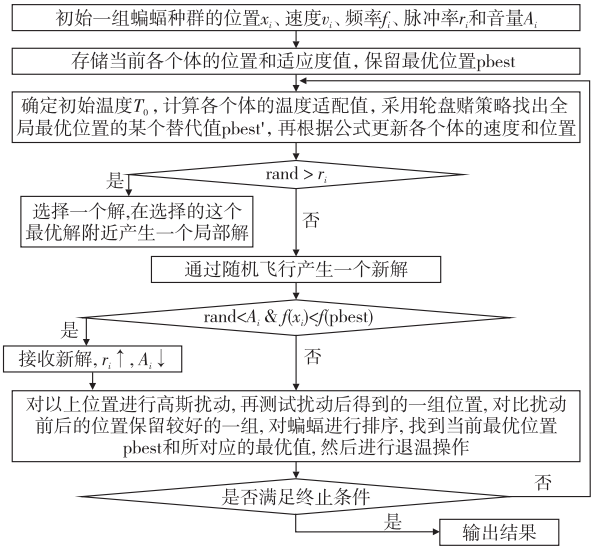

图 1　SAGBA 算法流程

4　仿真实验

为了研究 SAGBA 的有效性,本文进行了 BA、SAPSO[14] 和 SAGBA 在 20 个典型的基准测试函数中的优化性能比较,可以直观地看到 SAGBA 较好的优化性能。

4.1　实验设计

三种算法的参数设计如表 1 所示。

表 1　参数设计

算法	参数设计
BA	蝙蝠个数为 40,$\alpha = \gamma = 0.9$
SAPSO	粒子数为 40,学习因子 $c_1 = c_2 = 2.05$,退火常数为 0.5
SAGBA	蝙蝠个数为 40,$\alpha = \gamma = 0.9$,退火常数为 0.5

本文所用的 20 个典型基准测试函数均来自全局优化测试函数库[16],其中“ - - ”表示有很多个不同的最优点,如表 2 所示。

为了评价算法的收敛性能,要求最大迭代次数设为 2 000,连续运行 50 次,将测试函数的全局最小值的平均值作为算法优化性能的衡量指标,并采用 t-test 的方法[17] 比较分析各算法的性能。

表 2　基准测试函数

基准函数	最优点	最优值
Ackley	(0,0)	0
Beale	(3,0.5)	0
Bohachevsky	(0,0)	0
Booth	(1,3)	0
Branin	(9.42478,2.475)	0.3979
Dixon & Price	(1,…,1)	0
Goldstein & Price	(0,1)	3
Griewank	(0,0)	0
Hartmann	(0.1146,0.5556,0.8525)	-3.8628
Hump	(0.0898, -0.7126)	0
Levy	(1,1)	0
Matyas	(0,0)	0
Powell(d = 24)	(3, -1,0,1,…)	0
Rastrigin	(0,0)	0
Rosenbrock	(1,1)	0
Shubert	- -	-186.731
Sphere(d = 30)	(0,…,0)	0
Sum Squares(d = 20)	(0,…,0)	0
Trid(d = 6)	- -	-50
Zakharov	(0,0)	0

4.2　实验结果分析

实验 1　BA、SAPSO 和 SAGBA 的进化曲线对比

在相同的最大迭代次数下,对以上 20 个测试函数进行仿真模拟,如图 2 ~ 21 所示。以下进化曲线图横轴均表示进化次数,纵轴均表示平均适应度值的对数,即 log(适应度值)。

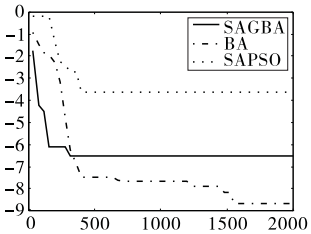

图 2　函数 Ackley 的进化曲线对比

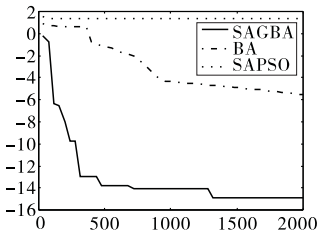

图 3　函数 Beale 的进化曲线对比

从图 2 中可以看出,对于 Ackley 函数来说,在进化初期 SAGBA 的个体质量就明显优于 SAPSO 和 BA 算法。随着迭代次数不断增加,SAPSO 算法在经过 400 次迭代后就已经陷入早熟收敛且无法跳出。BA 算法相较于 SAPSO 算法而言,其个体质量都有所提高,但其同样面临陷入早熟收敛的窘境,虽然在第 400、640、1 200 次等多次出现拐点和跳出局部最优的窘境,且收敛精度比较高,但其收敛太慢,迭代次数相对来说比较大。SAGBA 虽然在迭代过程中也同样出现过早熟收敛的现象,但在第 310 次迭代时出现了明显的拐点,之后在达到较高的收敛精度(10^{-7})和较少的迭代次数(310 次)情况下,达到最优解,表明该算法引导本已陷入局部最优值的个体偏离原先局部最优点,以较大概率向全局最优点靠近。

从图 3 中同样可以看出,对于 Beale 函数来说,在进化初期 SAGBA 的个体质量就明显优于 SAPSO 和 BA 算法。随着迭代次数不断增加,SAPSO 算法在经过 100 次迭代后就已经陷入早熟收敛且无法跳出,且并未收敛到全局最优点。BA 相较于 SAPSO 算法而言,其个体质量都有所提高,但其同样面临陷入早熟收敛的窘境,虽然在第 800 次迭代处出现拐点,跳出局部最优点,但其收敛精度和迭代次数并不可观。而 SAGBA 虽然在迭代过程中也同样出现过早熟收敛的现象,但在第 500 次迭

代时收敛到了全局最优点且达到了非常高的收敛精度。

从图 4 来看,与图 2 类似,对 Bohachevsky 函数来说,在进化初期 SAGBA 的个体质量就明显优于 SAPSO 和 BA 算法。随着迭代次数不断增加,虽然 SAPSO、BA 和 SAGBA 均在拐点处跳出局部最优点,以较大概率向全局最优点靠近,但是当迭代次数相同(如第 400 次)时,SAGBA 的收敛速度更快,收敛精度也相对比较高。

从图 5 中可以更容易地看出,对于 Booth 函数来说,只有 SAGBA 搜寻到了函数的全局最优点,SAPSO 算法陷入局部最优点,而 BA 并没达到收敛状态。

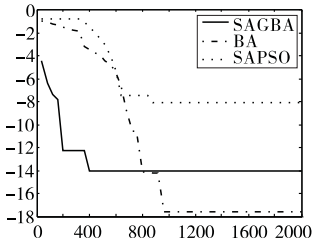

图 4　函数 Bohachevsky 的进化曲线对比

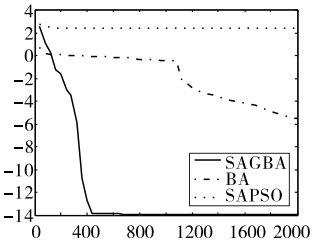

图 5　函数 Booth 的进化曲线对比

从图 6 可以看出,对于 Branin 函数来说,虽然在进化初期 BA 的个体质量要优于 SAGBA 和 SAPSO 算法,但是 SAPSO 算法陷入局部最优点而无法跳出,BA 并没有搜寻到全局最优点,只有 SAGBA 收敛到了全局最优点。

从图 7 中可以看出,对于 Dixon & Price 函数来说,三种算法的搜索能力与图 5、6 类似,只有 SAGBA 收敛到了全局最优点,SAPSO 算法陷入局部最优点而无法跳出,BA 并没有搜寻到全局最优点。

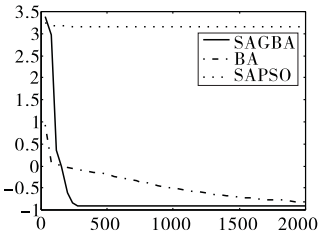

图 6　函数 Branin 的进化曲线对比

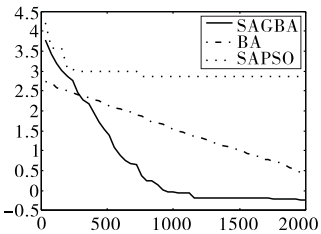

图 7　函数 Dixon & Price 的进化曲线对比

从图 8 中可以看出,对于 Goldstein & Price 函数来说,SAPSO 算法的搜索能力太差,根本没有搜寻到该函数的极值点,BA 和 SAGBA 均能收敛到函数的全局最优点,但是 SAGBA 的收敛速度更快,收敛到全局最优点所用迭代次数更少些。

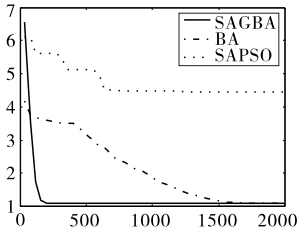

图 8　函数 Goldstein & Price 的进化曲线对比

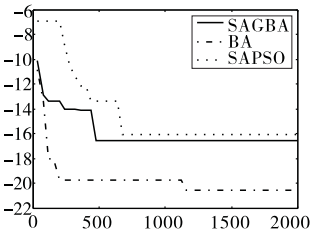

图 9　函数 Griewank 的进化曲线对比

从图 9 中可以看出,对于 Griewank 函数来说,在进化初期 SAGBA 的个体质量要优于 SAPSO 而差于 BA。显然 SAGBA 在整体进化过程中,无论是收敛速度还是收敛精度均要优于 SAPSO 算法,但是与 BA 相比,其收敛精度并不占优势。不过由于 BA 在进行到第 200 次和第1 190次迭代出现拐点,以跳出局部最优点,从而说明在稳定性上 BA 并不优于 SAGBA。

从图 10 中可以看出,对于 Hartmann 函数来说,在进化初期 SAGBA 的个体质量就明显优于 SAPSO 和 BA 算法。随着迭代次数的不断增加,SAPSO 算法并未收敛到全局最优点。BA 同样面临陷入早熟收敛的窘境,虽然在第 1 700 次迭代处出现拐点,收敛到全局最优点,但其收敛速度相对来说并不好。而 SAGBA 虽然在迭代过程中也同样出现过早熟收敛的现象,但在第 100 次迭代时出现了明显的拐点,以较大概率收敛于全局最优点。

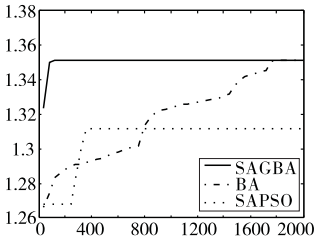

图 10　函数 Hartmann 的进化曲线对比

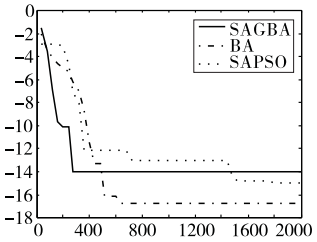

图 11　函数 Hump 的进化曲线对比

从图 11 中可以看出,对 Hump 函数来说,虽然 SAPSO、BA 和 SAGBA 均在拐点处跳出局部最优点,以较大概率向全局最优点靠近,但是当迭代次数相同时(如第 300 次),SAGBA 的收敛速度更快,收敛精度也相对较高。

从图 12 来看,对 Levy 函数来说,三种算法均可以搜寻到全局最优点,但是 SAGBA 比 SAPSO 算法收敛精度更高,收敛速度也更快,比 BA 更稳定。

从图 13 中仍然可以看出,对于 Matyas 函数来说,虽然 SAPSO、BA 和 SAGBA 均在拐点处跳出局部最优点,以较大概率向全局最优点靠近,但是当迭代次数相同(如第 300 次)时,SAGBA 的收敛速度更快,收敛精度也相对较高。

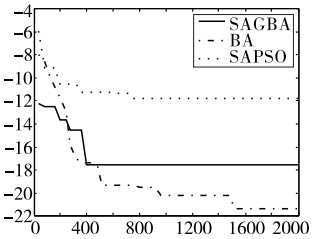

图 12　函数 Levy 的进化曲线对比

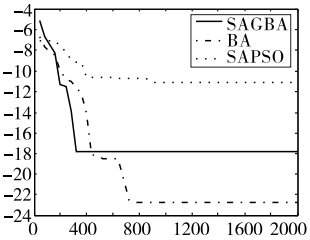

图 13　函数 Matyas 的进化曲线对比

从图 14 中可以看出,对于 Powell 函数来说,SAPSO 和 BA 算法并没有收敛到全局最优点,只有 SAGBA 在最大迭代次数的前提下收敛到全局最优点。

由于图 15 与图 13 类似,对于 Rastrigin 函数来说,可以得到相同的结论:在迭代次数相同(如第 100 次)时,SAGBA 的收敛速度更快,收敛精度也相对较高。

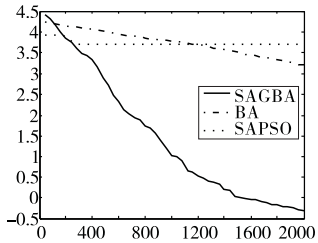

图 14　函数 Powell 的进化曲线对比

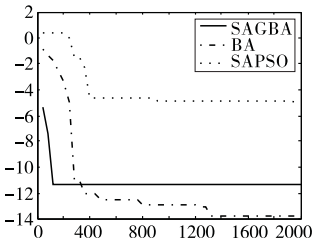

图 15　函数 Rastrigin 的进化曲线对比

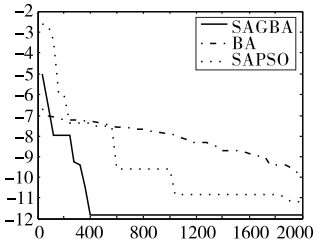

图 16　函数 Rosenbrock 的进化曲线对比

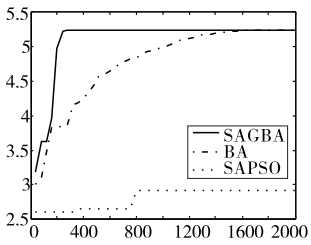

图 17　函数 Shubert 的进化曲线对比

从图 16 中可以看出,对于 Rosenbrock 函数来说,不论是进化前期还是后期,SAGBA 均优于其他两种算法。对 SAPSO 和 BA 算法来说并没有收敛到全局最优点,只有 SAGBA 以较快的收敛速度收敛到了全局最优点。

从图 17 中可以看出,对于 Shubert 函数来说,在最大迭代次数为 2 000 的限制条件下,三种算法均未收敛到函数的全局最优点,但是 SAGBA 的收敛速度是最快的。

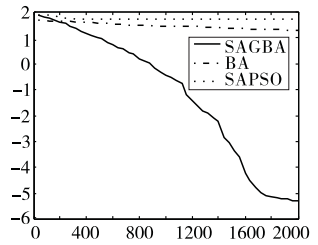

图 18　函数 Sphere 的进化曲线对比

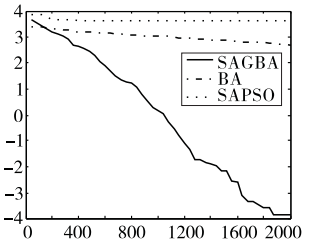

图 19　函数 Sum Squares 的进化曲线对比

图 18 与图 19 类似,所以仅分析图 19。对于 Sum Squares 函数来说,SAPSO 算法和 BA 的个体均陷入局部最优点而无法跳出,只有 SAGBA 可以在较大的范围内进行较快的搜索且收敛精度最大。

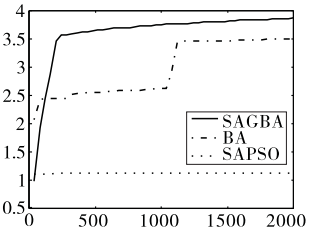

图 20　函数 Trid 的进化曲线对比

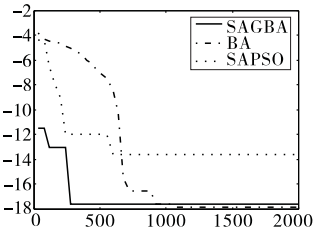

图 21　函数 Zakharov 的进化曲线对比

从图 20 可以看出,对于 Trid 函数来说,在最大迭代次数为 2 000次的限制条件下,三种算法均未收敛到函数的全局最优点,但是 SAGBA 的收敛速度是最快的且最接近全局极值点。

从图 21 中可以看出,对于 Zakharov 函数来说,不管是进化前期还是进化后期,SAGBA 的个体质量均优于 SAPSO 和 BA 算法,且可以发现,在迭代次数相同(如第 400 次)时,SAGBA 的收敛速度和收敛精度均要高于其他两种算法。

纵观所有函数的进化曲线对比,可以得到以下结论:SAGBA 比其他两种算法的收敛速度快,搜索精度更优,即基于模拟退火的高斯扰动蝙蝠优化算法的收敛性要优于其他两种算法的收敛性能。

实验 2　BA、SAPSO 和 SAGBA 的全局平均最优值对比

为了进一步研究 SAGBA 的优化性能,在同等实验条件下,设置最大迭代次数为 2 000 次,独立运行 50 次,取这 50 次的平均值进行比较,其中“ - - ”表示在最大迭代次数为 2 000 次的 50 次独立实验中未找到最小值。为了使图表清晰明了,现特用 Boh 表示 Bohachevsky 函数,Dixon 表示 Dixon&Price 函数,Goldstein 表示 Goldstein&Price 函数,如表 3 所示。

表 3　四种算法的全局平均最优值对比

测试函数	全局平均最优值		
	BA	SAPSO	SAGBA
Ackley	0.00018001	0.00156837	6.1879e - 005
Beale	0.00228013	1.09882604	1.4105e - 008
Boh	1.4737e - 007	0.00022725	1.1597e - 011
Booth	5.5471e - 008	—	9.8286e - 007
Branin	0.77166193	—	0.39789001
Dixon	—	—	5.5960e - 004
Goldstein	3.03614722	—	3.00003207
Griewank	1.4375e - 010	5.7773e - 008	3.6363e - 008
Hartmann	- 3.86261952	- 3.77258017	- 3.86272212
Hump	0.00050613	5.0153e - 008	6.1522e - 010
Levy	2.5251e - 009	3.5721e - 006	1.7607e - 010
Matyas	3.0342e - 010	2.5055e - 008	3.7909e - 009
Powell	—	—	0.003741511
Rastrigin	—	0.00019425	1.5840e - 007
Rosenbrock	0.00078735	0.00353827	4.6779e - 006
Shubert	- 186.730931	—	- 186.7297027
Sphere	0.00191766	0.04860173	4.6101e - 005
Sum Squares	—	—	3.6016 e - 004
Trid	- 12.7691163	- 3.50747926	- 49.5717002
Zakharov	9.2307e - 009	8.7796 e - 004	6.1224e - 010

表 3 给出了 BA、SAPSO 和 SAGBA 在 20 个测试函数中的计算结果。从表中结果可以看出,在函数 Booth 和 Matyas 上,SAGBA 得到的结果不如 BA 得到的结果,但优于 SAPSO 算法得到的结果;在函数 Griewank 上,SAGBA 所得到的结果要差于 BA 所得到的结果,与 SAPSO 算法得到的结果差不多;在函数 Goldstein&Price、Hartmann 和 Shubert 上,SAGBA 得到的结果与 BA 得到的结果差不多,对于除此之外的函数,SAGBA 的计算结果均要优于 BA 和 SAPSO 算法。即基于模拟退火的高斯扰动蝙蝠优化算法的搜索能力均优于其他两种优化算法的搜索能力。

实验 3　统计检验

由于每种算法在每个测试函数上都是独立运行 50 次,因而在每个测试函数问题上,每种算法都有 50 个样本。本文采用 T 检验法比较分析 SAGBA 与 BA、SAPSO 算法的性能。该 T 检验使用的是自由度为 58,显著水平为 0.05 的双尾检验。其结果显示在表 4 中。

表 4　BAGBA 与其他两种算法的 t 检验

测试函数	BA	SAPSO
Ackley	S +	S +
Beale	S +	S +
Bohachevsky	S +	S +
Booth	~	S +
Branin	S +	S +
Dixon & Price(d = 25)	S +	S +
Goldstein & Price	S +	S +
Griewank	S -	~
Hartmann	~	~
Hump	S +	S +
Levy	S +	S +
Matyas	~	~
Powell(d = 24)	S +	S +
Rastrigin	S +	S +
Rosenbrock	S +	S +
Shubert	~	S +
Sphere(d = 30)	S +	S +
Sum Squares(d = 20)	S +	S +
Trid(d = 6)	S +	S +
Zakharov	~	S +

表 4 中检验结果“S +”表示 SAGBA 在该行所对应的问题上的计算结果要显著性优于该列所对应的算法,“S -”表示该列所对应的算法在该行所对应的问题上显著性优于 SAGBA,“ ~”表示 SAGBA 在该行所对应的问题上的性能与该列所对应的算法的性能是相等的。

从表 4 中可以直观地看出,与 BA 相比,在 Griewank 函数上,SAGBA 的显著性差于 BA;在 Booth、Hartmann、Matyas、Shubert 和 Zakharov 函数上,SAGBA 的显著性均与 BA 相同;除此之外,对于其他的函数,SAGBA 的显著性均要优于 BA。与 SAPSO 算法相比,除了在 Griewank、Hartmann 和 Matyas 上 SAGBA 的显著性与 SAPSO 算法相同以外,在其他的函数上,SAGBA 的显著性均优于 SAPSO 算法。从统计的角度可以得出以下结论:SAGBA 在大部分的测试问题上都显著性优于 BA 和 SAPSO 算法。

5　结束语

将模拟退火算法与蝙蝠优化算法相结合,对蝙蝠个体进行高斯扰动,进一步搜索以保留个体“精英”。随着进化过程的推进,温度逐渐降低,接收较差解的几率逐渐减小,从而提高算法的收敛性能。仿真实验结果表明,该算法的收敛性能在不同程度上优于其他两种算法。下一步将研究 SAGBA 与其他优化算法的比较,并进一步研究 SAGBA 在多目标规划问题中的应用。

参考文献:

[1] YANG Xin-she. A new met heuristic bat-inspired algorithm [C]// Nature Inspired Cooperative Strategies for Optimization. 2010: 65-74.

[2] YANG Xin-she. Bat algorithm for multiobjective optimization [J]. International Journal Bio-Inspired Computation, 2011, 3(5): 267-274.

[3] 李枝勇,马良,张惠珍. 遗传变异蝙蝠算法在 0-1 背包问题上的应用[J/OL]. [2012-10-11]. http://www. cnki. net/kcms/detail/11.2127. TP. 20121011. 1019. 027. html.

[4] LEMMA T A, BIN M H F. Use of fuzzy systems and bat algorithm for energy modeling in a gas turbine generator [C]//Proc of IEEE Colloquium on Humanities, Science and Engineering. 2011: 305-310.

[5] YANG Xin-she, GANDOMI A H. Bat algorithm: a novel approach for global engineering optimization [J]. Engineering Computations, 2012, 29(5): 464-483.

[6] MISHRA S, SHAW K, MISHRA D. A new metaheuristic classification approach for microarray data[J]. Procedia Technology, 2012, 4(1): 802-806.

[7] KHAN K, NIKOV A, SAHAI A. A fuzzy bat clustering method for ergonomic screening of office workplaces, S3T 2011[C]//Advances in Intelligent and Soft Computing. 2011: 59-66.

[8] KHAN K, SAHAI A. A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context [J]. International Journal of Intelligent Systems and Applications, 2012, 4(7): 23-29.

[9] ALTRINGHAM J D. Bats: biology and behaviour [M]. Oxford: Oxford University Press, 1996: 37-64.

[10] KENNEDY J, EBERHART R. Particle swarm optimization [C]// Proc of IEEE International Conference on Neural Networks. 1995: 47-53.

[11] 康立山. 非数值并行算法——模拟退火算法[M]. 北京:科学出版社,1997.

[12] 吴志远,邵惠鹤,吴新余. 遗传退火进化算法[J]. 上海交通大学学报,1997,31(12): 69-71.

[13] 王雪梅,王义和. 模拟退火算法与遗传算法的结合[J]. 计算机学报,1997,20(4): 381-384.

[14] 赵世安,黄敢基. 模拟退火并行粒子群优化算法程序设计与研究[J]. 百色学院学报,2006,19(6): 9-12.

[15] 龚纯,王正林. 精通 MATLAB 最优化计算[M]. 北京:电子工业大学,2009.

[16] HEDAR J. Test functions for unconstrained global optimization [EB/OL]. http://www-optima. amp. i. kyoto-u. ac. jp/member/student/hedar/Hedar_files/TestGO_files/Page364. htm.

[17] 龚鲁光. 概率论与数理统计[M]. 北京:清华大学出版社,2006.

(上接第 372 页)性进行区域划分,在区域内利用空间邻域关系和 R^*-树确定空间邻域,在此基础上提出了全息熵空间离群度的度量方法和基于全息熵的空间离群度的空间离群点挖掘算法。所提算法有效解决了混合属性的基于全息熵的空间离群度的度量方法。由于实现区域划分,算法可进行并行计算,从而适合大数据量的计算。理论和实验证明,本算法在计算效率和实验结果的可解释性方面均具有优势。

参考文献:

[1] 薛安荣. 空间离群点挖掘技术的研究[D]. 镇江:江苏大学,2008.

[2] 薛安荣,姚林,鞠时光,等. 离群点挖掘方法综述[J]. 计算机科学, 2008,35(11):13-18.

[3] SHEKHAR S, LU C T, ZHANG Pu-sheng, *et al*. A unified approach to spatial outliers detection[J]. GeoInformatica, 2003, 7(2): 139-166.

[4] CHAWLA S, SUN Pei. SLOM: a new measure for local spatial out-liers [J]. Knowledge and Information Systems, 2006, 9(4): 412-429.

[5] LU C T, CHEN De-chang, KOU Yu-feng. Detecting spatial outliers with multiple attributes[C]//Proc of the 15th Conference on Tool Artificial Intelligence. Washington DC: IEEE Computer Society, 2003: 122-128.

[6] AGGARWAL C C, YU P. Outlier detection for high dimensional data [C]//Proc of ACM SIGMOD International Conference on Management of Data. 2001:37-46.

[7] KELLER F, MÜLLER E, BÖHM K. HiCS: high contrast subspaces for density-based outlier ranking[C]//Proc of the 28th IEEE International Conference on Data Engineering. 2012: 1037-1048.

[8] WU Shu, WANG Sheng-rui. Information-theoretic outlier detection for large-scale categorical data[J]. IEEE Trans on Knowledge and Data Engineering, 2013, 25(3):589-602.

[9] COVER T M, THOMAS J A. Elements of information theory[M]. 2nd ed. New Jersey: Wiley & Sons, 2006.

[10] 薛安荣,鞠时光,何伟华,等. 局部离群点挖掘算法研究[J]. 计算机学报,2007,30(8): 1455-1463.