文章编号:1006-3080(2017)04-0525-08 **DOI**:10.14135/j.cnki.1006-3080.2017.04.011

一种基于椋鸟群行为的改进型蝙蝠算法

胡 飞, 孙自强
(华东理工大学化工过程先进控制和优化技术教育部重点实验室,上海 200237)

摘要:蝙蝠算法是一种新兴的元启发式算法,基本蝙蝠算法(BA)存在寻优精度低、易陷入局部最优等缺点。将椋鸟群的集体性行为引入到基本蝙蝠算法中,有效地提高了算法的搜索范围;引入线性递减权重,用于平衡全局搜索和局部搜索。通过一些测试函数对该算法进行仿真研究,结果表明改进的蝙蝠算法有效地避免了种群个体陷入局部最优,提高了算法的寻优精度,优化效果得到改善。

关键词:蝙蝠算法(BA); 椋鸟群行为; 权重; 局部最优
中图分类号:TP301.6 **文献标志码**:A

An Improved Bat Algorithm Based on Starling Flock Behavior

HU Fei, *SUN Zi-qiang*
(*Key Laboratory of Advanced Chemical Process Control and Optimization Technology, Ministry of Education, East China University of Science and Technology, Shanghai* 200237, *China*)

Abstract: Bat algorithm (BA) is a new metaheuristic algorithm. However, the standard BA has some shortcomings, e.g., low convergence precision and easily relapsing into the local optima. In this work, by introducing the collective behavior of the starling group into BA algorithm, the searching range of the standard BA algorithm can be effectively improved. Besides, a linear decreasing weight is introduced to balance the global search and the local search. Simulation results from Benchmark functions show that the improved algorithm can effectively avoid the local optimum and attain higher convergence precision.

Key words: bat algorithm (BA); starling group behavior; weight; local optima

蝙蝠算法是一种新兴的元启发优化算法,该算法可以看作是和声搜索算法和粒子群算法的混合[1]。目前,已有许多经典的优化算法应用于诸多方面。文献[2]将一种改进的混沌蜂群算法应用于流水车间调度,优化调度过程中的最大完成时间,优化效果较好。文献[3]针对化工过程的动态多目标问题,利用一种改进型粒子群算法对其进行优化,测试结果表明该算法能较好地解决工程问题。蝙蝠算法作为新兴的优化算法,同样具有较好的应用前景,已经广泛应用于工程和自然科学领域,如函数优化问题、生产调度问题以及多目标优化问题等。文献[4]将蝙蝠算法应用于一些工程函数寻优,结果表明该算法寻优精度较高。文献[5]将蝙蝠算法应用于混合流水车间调度问题,能够最小化车间调度的完工时间。文献[6]针对多目标优化问题提出了一种多目标蝙蝠算法,应用于一些多目标函数优化,寻优效果较佳。然而蝙蝠算法亦具有粒子群等优化算法的通病,即

收稿日期:2016-11-01
作者简介:胡 飞(1992-),男,安徽人,硕士生,研究方向为智能优化算法。E-mail:xueshanfeihu1992@163.com
通信联系人:孙自强,E-mail:sunziqiang@ecust.edu.cn

收敛速度慢、容易陷入局部最优、搜索精度较低。为了提高蝙蝠算法的控制效果,许多研究者对蝙蝠算法开展研究,并对其进行改进。刘长平等[7]提出基于 Lévy 飞行特征的蝙蝠算法,从算法仿生原理入手,采用 Lévy 飞行搜索策略,更真实地模拟蝙蝠的捕食行为。Topal 等[8]提出一种动态虚拟蝙蝠算法,将蝙蝠种群分成探索类和开拓类子种群,当探索类蝙蝠进行对外搜索最优值时,开拓类蝙蝠进行局部搜索,寻找最优值。李枝勇等[9]将量子计算引入到蝙蝠算法中,利用量子位对蝙蝠的位置进行编码,利用量子旋转门进行蝙蝠最优位置的搜索。Gandomi 等[10]提出具有混沌搜索策略的蝙蝠算法,并利用不同的混沌函数产生的混沌序列对精英个体进行混沌优化,验证了混沌策略的有效性。Niknam 等[11]提出了一种自适应蝙蝠算法,将自适应方法与蝙蝠算法结合,提高算法的种群多样性及搜索能力,并应用于电力系统中 UC 问题,能够快速解决 UC 问题。Wang 等[12]提出将差分进化算法与蝙蝠算法相结合,并用于解决 UCVA 三维路径规划问题,结果表明控制效果较好。

当前针对蝙蝠算法的改进主要有两个方面:(1)针对蝙蝠算法自身机理存在的变量进行改进,包括参数自适应、辅助参数的引入以及搜索策略等;(2)将其他的优化算法与蝙蝠算法相结合,提高基本蝙蝠算法的控制效果[13]。本文提出的改进策略属于第 1 种,将椋鸟群行为引入到蝙蝠算法中,当全局最优值陷入局部最优时,根据椋鸟群的集体性行为对蝙蝠的位置和速度进行更新。

1 基本蝙蝠算法

1.1 基本蝙蝠算法的位置和速度更新

将蝙蝠类比为在当前可行域内分布的搜索点,用目标函数值(适应度函数值)的大小来判断当前蝙蝠所处位置的优劣;将蝙蝠飞行移动探测目标的过程类比为搜索目标函数位置最优解的过程,则可用以下流程和公式来模拟蝙蝠回声定位的行为[14]。初始时刻,需对蝙蝠的频率进行初始化,同时,蝙蝠在飞行过程中也需对其速度和位置进行更新,两者的更新公式如式(1)~ 式(3)所示。

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

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

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

其中:f_i 为第 i 个蝙蝠发出的声波频率;$f_{\max}$ 和 $f_{\min}$ 为频率的上下限;$\beta \in [0,1]$ 是一个服从均匀分布的随机值;v_i^t 和 v_i^{t-1} 分别为第 i 个蝙蝠在 t 和 $t-1$ 时刻的速度;x_i^t 和 x_i^{t-1} 分别为第 i 个蝙蝠在 t 和 $t-1$ 时刻的位置;x_* 为当前位置最优解。

在搜索过程中还需进行局部搜索,公式如下:

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

其中:x_{new} 为蝙蝠更新后的位置;x_{old} 为蝙蝠更新前的位置;$\varepsilon \in [-1,1]$ 为随机数;A^t 为整个蝙蝠种群在 t 时刻的平均响度。

1.2 响度和脉冲发射率

在蝙蝠搜寻目标猎物时,需对响度 A_i 和脉冲发射率 r_i 进行更新。初始阶段,A_i 较大,r_i 较小,搜寻到目标猎物后,A_i 减小,r_i 增大[15]。A_i 和 r_i 更新公式如下:

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

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

式中:α 和 γ 为常数,$0 < \alpha < 1$,$\gamma > 0$;r_i^0 为脉冲发射率的最大值。

2 改进的蝙蝠算法

2.1 椋鸟群行为

椋鸟是一种群体性种群,在日常生活中会产生一些壮观的景象。当遇到天敌时,其中一些个体会改变飞行方向,并影响周围个体,如此使得整个种群方向迅速改变,从而躲避天敌的捕食。

由于每个个体都会与周围的个体交流信息,因此由该个体发出的信息会很快传播至整个种群,这种行为称为椋鸟群行为。在椋鸟群中,椋鸟个体一般与其附近的 7 个个体进行信息交流,这 7 个个体再分别与其附近的 7 个个体进行信息交流,进而遍布整个种群,达到整个种群的信息共享。一项分析表明,6 个或 7 个个体组成一个群体交流网络将有效平衡整个种群凝聚性及个体交流,因此个体与其附近的 7 个个体进行信息交流足以影响整个种群的行为[16]。

2.2 引入椋鸟群行为的蝙蝠算法[17](SFBA)

基本蝙蝠算法在搜索过程中易陷入局部最优,缺乏种群多样性。将椋鸟群行为引入基本蝙蝠算法中,有助于增加种群多样性,避免陷入局部最优。当种群迭代结果发生停滞或更差超过上限值(count_limit)时,挑选出一定数目(max_num)适应度值较差的个体,对这些个体的位置和速度进行变换,挑选出全局最优值,替换原先的种群全局最优值。

针对位置变换，依据式(7)对粒子 i 的位置进行更新。

$$x_i^{t\prime} = x_i^t + r_1\left(\frac{1}{7}\sum_{n\in N_k} x_n\right) \tag{7}$$

式中：$x_i^{t\prime}$ 为更新后的位置；r_1 为 $-1\sim1$ 的随机值；N_k 为个体 i 附近的 7 个个体的索引编号。针对个体 i，将依据欧几里得距离计算公式找出其附近的 7 个个体。计算出来的新位置将替换原来位置成为个体最优位置值。

针对速度变换，依据式(8)对粒子 i 的速度进行更新。

$$v_i^{t\prime} = v_i^t + r_2\left(\frac{1}{7}\sum_{n\in N_k} v_n\right) \tag{8}$$

式中：$v_i^{t\prime}$ 为更新后的速度；r_2 为 $0\sim1$ 的随机值。依据欧几里得距离计算公式找出粒子 i 附近的 7 个个体，将更新后的速度值替换更新前速度，成为个体最优值。

2.3　针对位置和速度引入权重

由于固定的惯性权重控制精度不高，控制效果较差，本文针对蝙蝠算法速度和位置更新公式，分别引入权重 w_1 和 w_2，见式(9)和式(10)。将权重 w_1 由固定权重变为线性递减权重。起初权重较大，加强全局搜索，随着迭代次数的增加，迭代后期，权重减小，有利于提高局部搜索能力。线性递减的权重公式如下：

$$v_i^t = w_1 v_i^{t-1} + (x_i^{t-1} - x_*) f_i \tag{9}$$

$$x_i^t = w_2 x_i^{t-1} + v_i^t \tag{10}$$

$$w_1 = w_{\max} - \frac{(w_{\max} - w_{\min})\mathrm{Iter}}{\mathrm{Iter_max}} \tag{11}$$

其中：$w_{\max}$ 为 w_1 的上限值；$w_{\min}$ 为 w_1 的下限值；Iter 为当前迭代次数；Iter_max 为最大迭代次数。

2.4　SFBA 算法流程

(1) 确立目标函数 $f(x)$，初始化算法的基本参数，包括种群大小 Pop_Size、声波频率 f 和声波频率的上下限[$f_{\max}$，$f_{\min}$]、Iter_max、种群的位置 x 和速度 v、最大响度 A_i 及其控制参数 α、最大脉冲发射率 r_i^0 初始值及其控制参数 γ、全局最优值停滞计数器上限值 count_limit。

(2) 对所有蝙蝠个体计算适应度函数值 $f(x)$，找出当前位置最优解 x_*。

(3) 按照式(1)随机产生频率的新解。

(4) 根据式(9)和式(10)对蝙蝠的速度和位置进行更新，针对速度设置的权重 w_1 使用线性递减权重，见式(11)。

(5) 产生一个随机值 rand，若 rand 大于脉冲发射率 r_i，则根据式(4)，从处在最佳位置的蝙蝠 i 个体中选取一个解，对该位置最优解进行随机扰动，产生一个局部解 x_{new} 替代蝙蝠 i 的当前位置。

(6) 计算适应度函数值(目标函数值)；

(7) 若随机值 rand 小于响度 A_i 且更新位置后的目标函数值小于当前目标函数最优值，则接受这个位置新解，并增大脉冲发射率 r_i 和响度 A_i，见式(5)、式(6)。

(8) 对位置变动后的蝙蝠重新排列，找出当前位置最优解和适应度函数最优值。

(9) 判断当前适应度最优值是否大于或等于上一代适应度最优值，若是，则计数器 count+1(count 初始值设为 0)。

(10) 判断 count 是否大于上限值 count_limit，若是，则挑选出适应度较差的 max_num 个个体，利用式(7)与式(8)更新该部分个体位置和速度值，替换原先个体最优值，并从中找出全局最优值，作为当前种群全局最优值，并置 count=0。

(11) 当迭代次数小于最大迭代次数时，回到步骤(2)继续迭代搜索最优解。

(12) 算法迭代结束，输出位置全局最优解。

3　算法的参数选择

SFBA 算法参数主要包含 w_1，w_2，count_limit，max_num，其中 w_1 主要由 $w_{\max}$ 和 $w_{\min}$ 决定，因此需要选择的参数主要为 $w_{\max}$，$w_{\min}$，w_2，count_limit，max_num，这 5 个参数可以通过实验确定。以 Sphere 函数为基准进行实验，种群规模设为 100，迭代次数 Iter_max 为 500 次，分别运行 20 次，取 20 次优化结果的平均值作为参考值，确定权值 $w_{\max}$，$w_{\min}$ 以及 w_2 的最优值。

表 1 示出了 $w_{\max}$ 为 0.45、0.46、0.47、0.48、0.49、0.50、0.51、0.52、0.53、0.54、0.55，$w_{\min}=0.1$，$w_2=0.50$ 时的运行结果，可以看出，$w_{\max}=0.50$ 时算法控制效果较好。

表 2 示出了 $w_{\min}$ 为 0.05、0.06、0.07、0.08、0.09、0.10、0.11、0.12、0.13、0.14、0.15，$w_{\max}=0.50$，$w_2=0.50$ 时的运行结果，可以看出，$w_{\min}=0.10$ 时算法控制效果较好。

表 3 示出了 w_2 为 0.45、0.46、0.47、0.48、0.49、0.50、0.51、0.52、0.53、0.54、0.55，$w_{\max}=0.50$，$w_{\min}=0.10$ 时的运行结果，可以看出，$w_2=0.50$ 时算法控制效果较好。

表1 w_{max}参数取值表

Table 1 Parameter value table of w_{max}

w_{max}	函数平均值
0.45	$5.893\,3\times10^{-13}$
0.46	$4.770\,9\times10^{-15}$
0.47	$1.493\,0\times10^{-17}$
0.48	$8.342\,4\times10^{-18}$
0.49	$5.387\,5\times10^{-18}$
0.50	$2.308\,5\times10^{-18}$
0.51	$4.489\,3\times10^{-18}$
0.52	$5.948\,3\times10^{-17}$
0.53	$9.153\,1\times10^{-15}$
0.54	$6.043\,9\times10^{-14}$
0.55	$8.558\,3\times10^{-10}$

表2 w_{min}参数取值表

Table 2 Parameter value table of w_{min}

w_{min}	函数平均值
0.05	$9.520\,8\times10^{-9}$
0.06	$4.402\,0\times10^{-14}$
0.07	$3.635\,9\times10^{-15}$
0.08	$1.849\,3\times10^{-17}$
0.09	$1.256\,8\times10^{-18}$
0.10	$4.024\,8\times10^{-19}$
0.11	$6.480\,0\times10^{-18}$
0.12	$6.195\,2\times10^{-18}$
0.13	$8.632\,5\times10^{-17}$
0.14	9.3657×10^{-14}
0.15	$5.202\,6\times10^{-13}$

表3 w_2参数取值表

Table 3 Parameter value table of w_2

w_2	函数平均值
0.45	$7.942\,0\times10^{-11}$
0.46	$4.235\,0\times10^{-14}$
0.47	$3.404\,3\times10^{-15}$
0.48	$3.192\,2\times10^{-17}$
0.49	$2.348\,3\times10^{-17}$
0.50	$1.479\,5\times10^{-18}$
0.51	$3.322\,1\times10^{-18}$
0.52	$6.839\,6\times10^{-17}$
0.53	$1.650\,4\times10^{-17}$
0.54	$8.943\,5\times10^{-13}$
0.55	$8.248\,2\times10^{-12}$

表4示出了count_limit为1、2、3、4、5时的运行结果,可以看出,count_limit=3时算法控制效果较好。

表5示出了max_num为13、17、19、22、23时的运行结果,可以看出,max_num=19时算法控制效果较好。

表4 count_limit参数取值表

Table 4 Parameter value table of count_limit

count_limit	函数平均值
1	$3.039\,5\times10^{-15}$
2	$1.403\,9\times10^{-18}$
3	$7.356\,8\times10^{-19}$
4	$4.346\,5\times10^{-17}$
5	$9.452\,0\times10^{-16}$
6	$5.394\,9\times10^{-13}$

表5 max_num参数取值表

Table 5 Parameter value table of max_num

max_num	函数平均值
13	$1.953\,2\times10^{-10}$
17	$5.030\,4\times10^{-17}$
19	$2.209\,5\times10^{-18}$
22	$8.420\,2\times10^{-15}$
23	$3.405\,0\times10^{-11}$

综上所述,对于改进型蝙蝠算法,取$w_{max}=0.50$,$w_{min}=0.10$,$w_2=0.50$,count_limit=3,max_num=19,采用这组参数进行算法测试,可以进一步验证改进型蝙蝠算法的可行性。

4 测试结果

为了评估本文提出的改进型蝙蝠算法的寻优效果,选择5个标准测试函数(如表6所示)进行测试。

基本蝙蝠算法和改进型蝙蝠算法的种群数均为100,迭代次数为500次。基本蝙蝠算法中,蝙蝠个体的最大脉冲响度$A_i=0.25$,最大脉冲发射率$r_i^0=0.75$,响度控制参数$\alpha=0.95$,脉冲发射率控制参数$\beta=0.95$,频率f取值范围为[0,2]。改进型蝙蝠算法中,最大脉冲响度A_i、最大脉冲发射率r_i^0、响度控制参数α及脉冲发射率控制参数β与基本蝙蝠算法一致,权值$w_{max}=0.50$,$w_{min}=0.10$,$w_2=0.50$,count_limit=3,max_num=19。BA和SFBA在维

度 d 为 10 维和 20 维分别计算 20 次，记录 20 次的最优值、最差值并计算 20 次结果的平均值和标准差，结果如表 7 所示。

表 6　标准测试函数

Table 6　Benchmark functions

函数名称	函数解析式	自变量范围	理论收敛值
Ackley	$f_1(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)]+20+\mathrm{e}$	$[-32,32]$	0
Griewank	$f_2(x)=\frac{1}{4\,000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]$	0
Rastrigin	$f_3(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]$	0
Rosenbrock	$f_4(x)=\sum_{i=1}^{n-1}[100\,(x_i^2-x_{i+1})^2+(x_i-1)^2]$	$[-30,30]$	0
Sphere	$f_5(x)=\sum_{i=1}^{n}x_i^2$	$[-100,100]$	0

表 7　标准测试函数寻优结果对比

Table 7　Comparison of Benchmark functions optimization results

函数	维度	算法	最优值	最差值	平均值	标准差
f_1	10	BA	$6.337\,0\times10^{-1}$	1.796 9	1.411 3	$2.679\,1\times10^{-1}$
		SFBA	$1.759\,5\times10^{-12}$	$7.559\,0\times10^{-9}$	$8.022\,5\times10^{-10}$	$1.676\,1\times10^{-9}$
	20	BA	1.580 4	3.160 6	2.300 2	$4.325\,2\times10^{-1}$
		SFBA	$2.804\,0\times10^{-12}$	$7.426\,5\times10^{-9}$	$1.158\,5\times10^{-9}$	$1.997\,4\times10^{-9}$
f_2	10	BA	$1.284\,5\times10$	$5.917\,5\times10$	$3.554\,5\times10$	$1.059\,5\times10$
		SFBA	0	0	0	0
	20	BA	$1.256\,2\times10^{2}$	$2.155\,4\times10^{2}$	$1.723\,0\times10^{2}$	$2.542\,1\times10$
		SFBA	0	0	0	0
f_3	10	BA	$1.231\,4\times10$	$3.805\,7\times10$	$2.303\,5\times10$	7.990 9
		SFBA	0	0	0	0
	20	BA	$4.726\,7\times10$	$1.219\,1\times10^{2}$	$7.511\,9\times10$	$1.846\,7\times10$
		SFBA	0	0	0	0
f_4	10	BA	7.938 9	$2.774\,6\times10^{3}$	$4.033\,8\times10^{2}$	$7.103\,2\times10^{2}$
		SFBA	8.941 9	9.000 0	8.988 9	$1.656\,1\times10^{-2}$
	20	BA	$2.734\,0\times10$	$1.555\,3\times10^{3}$	$2.889\,2\times10^{2}$	$4.424\,0\times10^{2}$
		SFBA	$1.893\,5\times10$	$1.900\,0\times10$	$1.899\,1\times10$	$1.683\,7\times10^{-2}$
f_5	10	BA	$2.800\,0\times10^{-2}$	$7.430\,0\times10^{-2}$	$5.700\,5\times10^{-2}$	$1.117\,1\times10^{-2}$
		SFBA	$2.032\,5\times10^{-22}$	$1.739\,1\times10^{-17}$	$1.114\,1\times10^{-18}$	$3.787\,7\times10^{-17}$
	20	BA	$2.943\,0\times10^{-1}$	1.282 4	$4.847\,7\times10^{-1}$	$2.077\,1\times10^{-1}$
		SFBA	$4.268\,6\times10^{-22}$	$4.097\,0\times10^{-17}$	$2.708\,7\times10^{-18}$	$2.382\,2\times10^{-17}$

测试函数 f_1,f_2,f_3 均为多峰值测试函数,BA寻优精度较低,控制效果较差,而SFBA均能取得较高的寻优精度,控制效果较好。测试函数 f_4 是一种典型的病态非凸单模函数,难以极小化,从表7中数据可以看出BA和SFBA寻优效果均不理想。测试函数 f_5 为简单的单峰值函数,在自变量取值范围内仅有一个全局极小值点,因而BA的寻优效果还可以,但是相比之下,SFBA的寻优精度更高,控制效果更好。限于篇幅,本文仅给出部分维度的寻优效果,测试函数迭代曲线对比如图1~图5所示。

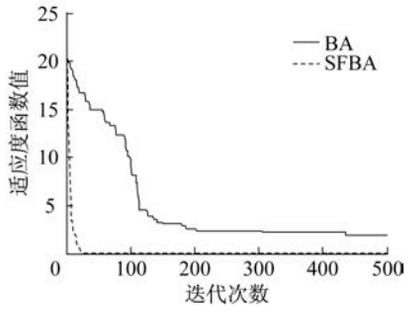


图1 测试函数 f_1 迭代曲线对比($d=20$)

Fig. 1 Comparison of Benchmark function f_1's iterative curves ($d=20$)

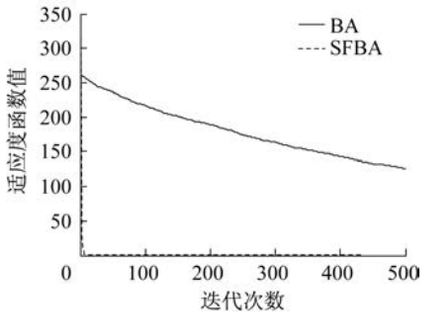


图2 测试函数 f_2 迭代曲线对比($d=20$)

Fig. 2 Comparison of Benchmark function f_2's iterative curves ($d=20$)

为了进一步验证本文算法的可行性,将SFBA与文献[18]基于微分算子与Lévy飞行策略的蝙蝠算法(DLBA)及文献[19]基于遗传交叉因子的蝙蝠算法(GHBA)进行对比,结果如表8所示。从表8中数据可知,SFBA的寻优效果优于GHBA、DLBA。

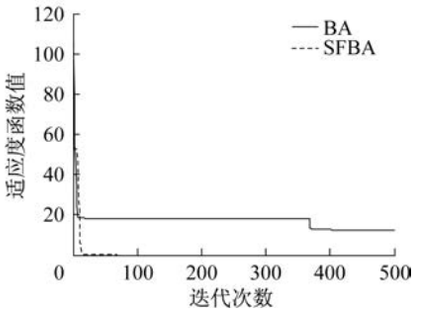


图3 测试函数 f_3 迭代曲线对比($d=10$)

Fig. 3 Comparison of Benchmark function f_3's iterative curves ($d=10$)

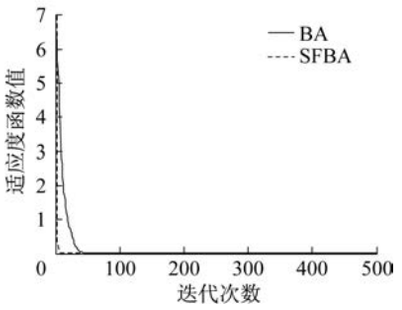


图4 测试函数 f_4 迭代曲线对比($d=10$)

Fig. 4 Comparison of Benchmark function f_4's iterative curves ($d=10$)

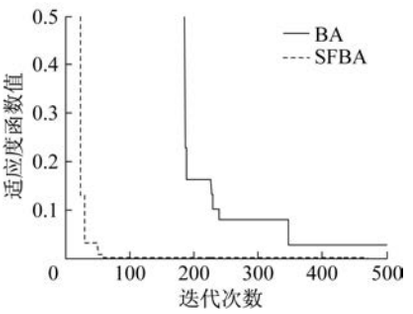


图5 测试函数 f_5 迭代曲线对比($d=10$)

Fig. 5 Comparison of Benchmark function f_5's iterative curves ($d=10$)

表8 改进的BA算法在5个测试函数寻优结果对比

Table 8 Comparison of improved BA algorithms in 5 Benchmark function optimization results

函数	SFBA		GHBA		DLBA	
	最优值	平均值	最优值	平均值	最优值	平均值
f_1	$2.804\ 0\times10^{-12}$	$1.158\ 5\times10^{-9}$	1.31×10^{-7}	3.41×10^{-2}	$4.717\ 4\times10^{-7}$	6.4097×10^{-6}
f_2	0	0	3.65×10^{-7}	1.41×10^{-5}	$1.381\ 8\times10^{-7}$	$5.010\ 3\times10^{-6}$
f_3	0	0	0	0	$5.999\ 8\times10^{-8}$	$4.470\ 5\times10^{-6}$
f_4	$1.893\ 5\times10$	$1.899\ 1\times10$	7.64×10^{-5}	1.51	5.667 2	7.390 3
f_5	$4.268\ 6\times10^{-22}$	$2.708\ 7\times10^{-18}$	8.08×10^{-19}	5.65×10^{-18}	$4.305\ 7\times10^{-8}$	$4.803\ 6\times10^{-6}$

5　改进型蝙蝠算法在 PID 参数整定中的应用

为了进一步验证本文算法的有效性，将改进型蝙蝠算法应用于 PID 控制器参数优化中[20]。选取二阶被控对象，传递函数如下：

$$G(s)=\frac{400}{60s^2+50s} \tag{12}$$

采样时间为 2 ms，输入一个阶跃信号，采用误差绝对值时间积分性能指标作为参数选择的最小目标函数，为防止能量过大，在目标函数中加入控制输入平方项，最优指标如式(13)所示。

$$J=\int_0^{\infty}[w_1|e(t)|+w_2u^2(t)]\mathrm{d}t+w_3t_u \tag{13}$$

式中：J 表示目标函数；$e(t)$表示系统误差；$u(t)$为控制器输出；t_u 为上升时间；w_1，w_2，w_3 为权重值。

为避免超调，采取惩罚功能，一旦出现超调，就将超调量作为最优指标中的一项，此时最优指标如式(14)所示。

$$J=\int_0^{\infty}[w_1|e(t)|+w_2u^2(t)+w_4|ey(t)|]\mathrm{d}t+w_3t_u \quad \text{if } ey(t)<0 \tag{14}$$

其中：w_4 为权重值，且 $w_4>>w_1$；$ey(t)=y(t)-y(t-1)$，$y(t)$为被控对象输出。

SFBA 算法中，种群数为 100，$w_{max}=0.50$，$w_{min}=0.10$，$w_2=0.5$，count_limit=3，max_num=19，PID 参数 K_p，K_i，K_d 取值范围分别为[0，40]，[0，4]，[0，4]，取 $w_1=0.999$，$w_2=0.001$，$w_3=100$，$w_4=2.0$。迭代 100 次后，用 BA 算法优化 PID 三参数优化后 $K_p=31.711\ 6$，$K_i=3.354\ 7$，$K_d=0.147\ 2$，性能指标 $J=79.189\ 6$，适应度函数 J、K_p、K_i、K_d 的优化过程如图 6 所示。用 SFBA 算法优化 PID 三参数优化后 $K_p=35.053\ 4$，$K_i=3.527\ 6$，$K_d=0.003\ 1$，性能指标 $J=78.352\ 7$，适应度函数 J，K_p，K_i，K_d 的优化过程如图 7 所示。BA 和 SFBA 算法优化后，PID 控制阶跃响应如图 8 所示。

由图 8 可知，SFBA 算法优化效果优于 BA 算法，SFBA 基本无超调，调节时间较短，响应速度较快。

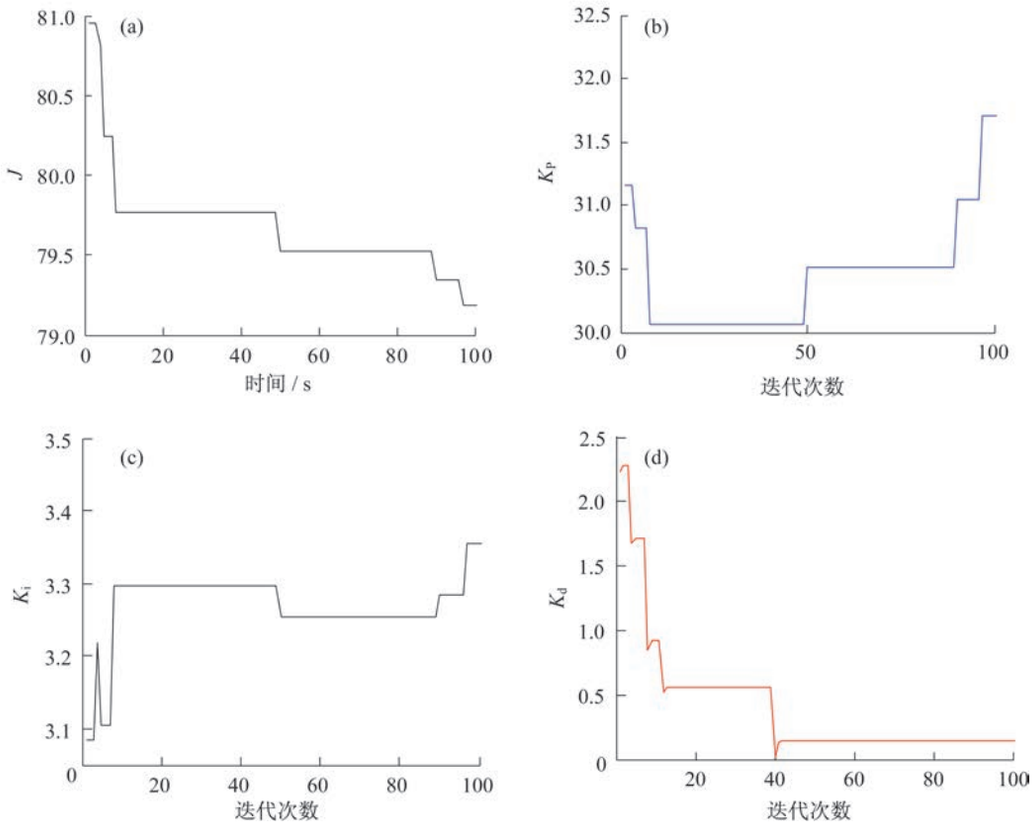


图 6　BA 算法迭代曲线

Fig. 6　Iterative curves of BA algorithm

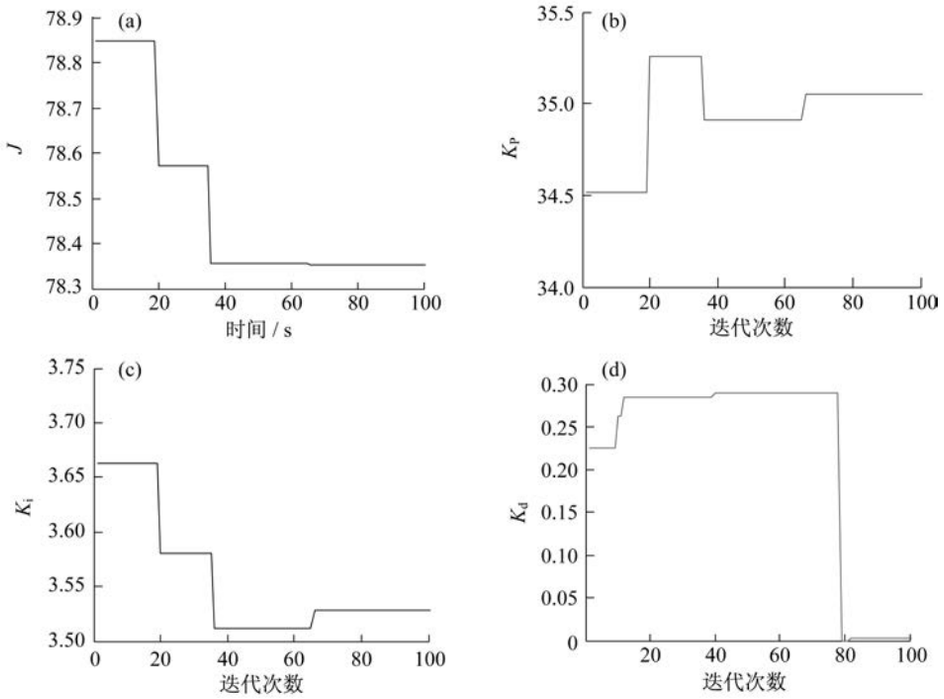


图 7 SFBA 算法的迭代曲线

Fig. 7 Iterative curves of SFBA algorithm

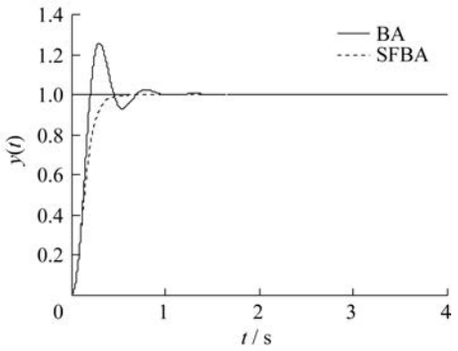


图 8 BA 算法与 SFBA 算法的阶跃响应曲线

Fig. 8 Unit-step response of BA algorithm and SFBA algorithm

6 结 论

本文针对 BA 算法收敛速度慢、容易陷入局部最优、搜索精度不高等缺点,提出了一种基于椋鸟群行为的蝙蝠算法(SFBA),同时引入线性递减权重平衡全局搜索与局部搜索。通过 5 个不同的单峰值和多峰值测试函数进行验证,与 BA 算法相比,SFBA 算法优化效果较佳。此外,将 SFBA 算法应用于 PID 参数优化,结果表明 SFBA 算法优化效果优于 BA 算法。

参考文献:

[1] YANG X S. A new metaheuristic bat-inspired algorithm[J]. Physics,2010,284:65-74.

[2] 刘华,顾幸生. 改进的混沌蜂群算法在流水线调度中的应用[J]. 华东理工大学学报(自然科学版),2013,39(3):345-350.

[3] 王珊珊,杜文莉,陈旭,等. 基于约束骨干粒子群算法的化工过程动态多目标优化[J]. 华东理工大学学报(自然科学版),2014,40(4):449-457.

[4] YANG X S, GANDOMI A H. Bat algorithm: A novel approach for global engineering optimization[J]. Engineering Computation,2012,29(5):267-289.

[5] MARICHELVAM M K, PRABAHARAN T, YANG X S, *et al*. Solving hybrid flow shop scheduling problems using bat algorithm[J]. International Journal of Logistics Economics & Globalisation,2013,5(1):15-29.

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

[7] 刘长平,叶春明. 具有 Lévy 飞行特征的蝙蝠算法[J]. 智能系统学报,2013(3):240-246.

[8] TOPAL A O, ALTUN O. A novel meta-heuristic algorithm: Dynamic virtual bats algorithm [J]. Information Sciences, 2016,354:222-235.

[9] 李枝勇,马良,张惠珍. 函数优化的量子蝙蝠算法[J]. 系统管理学报,2014(5):717-722.

(下转第 562 页)

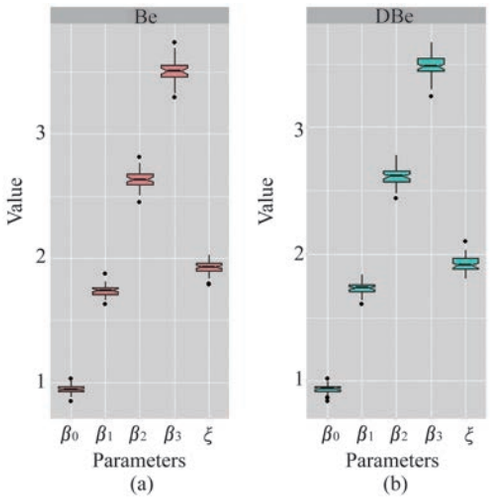


图1 分布式贝叶斯方法和传统方法参数估计精度对比分析结果

Fig. 1 Comparative analysis of distributed Bayes vs. classical method

4 结束语

本文针对不平衡二元响应变量的海量数据,基于广义极值分布进行回归分析,借助于分布式贝叶斯方法(Metropolish-Hastings抽样算法)进行参数估计,模拟研究表明该方法具有一定的计算优势,应用于模拟数据分析中取得了较好的效果。今后可以进一步推广到基于Gibbs抽样的分布式贝叶斯分析中。

参考文献:

[1] CZADO C, SANTNER T J. The effect of link misspecification on binary regression inference [J]. Journal of Statistical Planning and Inference, 1992, 33(2): 213-231.

[2] WANG X, DEY D K. Generalized extreme value regression for binary response data: An application to B2B electronic payments system adoption [J]. The Annals of Applied Statistics, 2010, 4(4): 2000-2023.

[3] STEVEN S L, BLOCKER A W, BONASSI F V. Bayes and big data: The consensus Monte Carlo algorithm [J]. International Journal of Management Science and Engineering Management, 2016, 11(2): 78-88.

[4] GHOSH S K, MUKHOPADHYAY P, LU J C. Bayesian analysis of zero-inflated regression models [J]. Journal of Statistical Planning and Inference, 2006, 136(4): 1360-1375.

[5] CHIPMA H A, GEORGE E I, MCCULLOCH R E. BART: Bayesian additive regression trees [J]. The Annals of Applied Statistics, 2010, 4(1): 266-298.

[6] DEAN J, GHEMAWAT S. Mapreduce: Simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-113.

(上接第532页)

[10] GANDOMI A H, YANG X S. Chaotic bat algorithm [J]. Journal of Computational Science, 2014, 5(2): 224-232.

[11] NIKNAM T, BAVAFA F, AZIZIPANAH-ABARGHOOEE R. New self-adaptive bat-inspired algorithm for unit comi--tment problem [J]. IET Science Measurement Technology, 2014, 8(6): 505-517.

[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: 231-238.

[13] AFRABANDPEY H, GHAFFARI M, MIRZAEI A, *et al*. A novel bat algorithm based on chaos for optimization tasks [C]//Iranian Conference on Intelligent Systems. USA: IEEE, 2014: 1-6.

[14] RAGHAVAN S, MARIMUTHU C, SARWESH P, *et al*. Bat algorithm for scheduling workflow applications in cloud [C]// International Conference on Electronic Design, Computer Networks & Automated Verification. USA: IEEE, 2015: 139-144.

[15] YıLMAZ S, KÜÇÜKSILLE E U. A new modification approach on bat algorithm for solving optimization problems [J]. Applied Soft Computing, 2014, 28(5): 259-275.

[16] YOUNG G F, SCARDOVI L, CAVAGNA A, *et al*. Starling flock networks manage uncertainty in consensus at low cost [J]. PloS Computational Biology, 2013, 9(1): e1002894.

[17] NETJINDA N, ACHALAKUL T, SIRINAOVAKUL B. Particle swarm optimization inspired by starling flock behavior [J]. Applied Soft Computing, 2015, 35(C): 411-422.

[18] XIE J, ZHOU Y, CHEN H. A novel bat algorithm based on differential operator and Lévy flights trajectory [J]. Computational Intelligence & Neuroscience, 2013(2013): 453-812.

[19] 彭泓,丁玉成. 基于遗传交叉因子的蝙蝠算法的改进[J]. 激光杂志, 2015(2): 23-26.

[20] 刘金琨. 先进PID控制MATLAB仿真[M]. 北京:电子工业出版社, 2004.