具有协同寻优的蝙蝠萤火虫混合优化算法*

温　泰,赵志刚**,莫海森
(广西大学计算机与电子信息学院,广西南宁　530004)

摘要:萤火虫算法存在着对于初始解分布的依赖性、后期收敛速度慢、易于停滞、早熟和求解精度低等缺陷。本研究在萤火虫算法引入蝙蝠种群在全局最优附近进行更加详细的局部搜索,以协助萤火虫种群进行寻优;并在寻优过程中加强蝙蝠种群与萤火虫种群的信息交互,协调寻优;最后对全局最优个体进行高斯扰动以增加种群的多样性,从而避免种群陷入局部最优解。通过使用6个常见的基准测试函数对该算法进行测试,并与其他3种算法(标准粒子群算法、蝙蝠算法、萤火虫算法)进行对比实验,结果表明该混合算法的总体性能优于其他3种算法。引入蝙蝠种群对萤火虫性能有较大提升,改善切实有效。

关键词:函数优化　蝙蝠算法　萤火虫算法　协同寻优

中图分类号:TP301.6　　**文献标识码**:A　　**文章编号**:1002-7378(2019)02-0140-07

0　引言

根据工程应用发展的需求,一系列群智能算法应运而生,如粒子群算法[1](Particle swarm optimization,PSO)通过模拟鸟群的觅食过程迭代寻优;蝙蝠算法[2](Bat algorithm,BA)是通过模仿蝙蝠声呐探测飞行的方式迭代寻优;萤火虫算法[3](Firefly algorithm,FA)通过模拟萤火虫在晚上的群聚活动进行寻优。萤火虫算法被广泛地运用于选址问题、预测模型、背包问题等工程应用问题。萤火虫算法作为一种新的群体智能仿生优化算法,发展时间尚短,算法本身存在着对于初始解分布的依赖性、后期收敛速度慢、易于停滞、早熟和求解精度低等缺陷[4]。近些年,相关学者对萤火虫算法进行了多角度的改进。Łuckasik等[5]于2009年对FA改进,并对算法的参数进行研究,改进后的FA提高了求解精度,但求解速度较慢。冯艳红等[6]提出了基于混沌理论的动态种群萤火虫优化算法(CDPFA),该算法运用立方映射混沌初始化萤火虫初始位置,取得了较好的效果,进一步提高了算法的寻优精度和求解速度。王吉权等[7]结合了遗传算法,提出一种基于目标函数自适应惯性权重的萤火虫算法。测试函数表明,通过此策略的改进,萤火虫算法个体之间的收敛速度可以得到有效地提高。Gandomi等[8]更是将混沌映射的理论应用到了萤火虫算法中,利用混沌理论对萤火虫的吸引度β和光吸收参数γ进行了改进,并用12种不同的混沌映射进行了仿真验证。与标准萤火虫算法比较,经过改进的萤火虫算法具有更高的鲁棒性和实用性。

*广西自然科学基金项目(2015GXNSFAA139296)资助。

【作者简介】
温　泰(1994—),男,硕士研究生,主要从事群智能算法优化研究。
【**通信作者】
赵志刚(1973—),男,博士,教授,主要从事智能优化计算研究,E-mail:zzgmail2002@163.com。

【引用本文】
DOI:10.13657/j.cnki.gxkxyxb.20190515.008
温泰,赵志刚,莫海森.具有协同寻优的蝙蝠萤火虫混合优化算法[J].广西科学院学报,2019,35(2):140-146.
WEN T,ZHAO Z G,MO H M. The hybrid bat and firefly algorithm with collaborative optimization [J]. Journal of Guangxi Academy of Sciences,2019,35(2):140-146.

付强等[9]根据改进的自调节步长的萤火虫搜索策略改善 PSO 的局部搜索能力,避免 PSO 陷入局部最小值。郝晓莹等[10]提出一种通过粒子群优化的粒子搜索,对不同优化问题自适应选取萤火虫算法参数值的优质组合的方法。萤火虫算法虽然有诸多的改进方法,但是总存在着些许不足,所以我们在萤火虫算法的基础上,引进了蝙蝠种群来协助萤火虫种群进行寻优,利用了蝙蝠算法具有较强的局部搜索能力这个特点,对全局最优个体附近进行详细的局部搜索;并加强蝙蝠种群和萤火虫种群在寻优过程中的信息交互,协同寻优;最后对全局最优进行高斯扰动以增加种群的多样性。该算法解决了萤火虫算法本身存在着对于初始解分布的依赖性、后期收敛速度慢、易于停滞、早熟和求解精度低等缺陷,可广泛地运用在解决工程应用问题或者人工智能领域,有效地提高工作效率与精度,为这些领域的研究应用建立基础。

1 材料与方法

1.1 蝙蝠算法

蝙蝠算法是根据蝙蝠在捕食的过程中利用回声来定位猎物的自然现象而提出的一种群智能算法[11]。蝙蝠利用回声的差异来判断猎物之间的差异,再根据猎物的接近程度自动调整他们发出脉冲的频率,通过不断的迭代从而寻找到猎物的位置。

蝙蝠以速度 v 随机飞行,并用不同的频率 Q 和音量 A 搜索猎物。标准蝙蝠算法通过公式(1)~(3)对蝙蝠个体的速度以及位置进行更新。

$$Q_i = Q_{\min} + (Q_{\max} - Q_{\min})\beta, \tag{1}$$

$$v_i^t = v_i^t + (X_i^{t-1} - X_{\text{Gbest}})Q_i, \tag{2}$$

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

其中,Q_i 是第 i 个蝙蝠飞行的频率,$Q_{\max}$ 和 $Q_{\min}$ 分别是频率取值范围的上限和下限;β 是[0,1]上服从均匀分布的随机向量;v_i^t 和 v_i^{t-1} 分别是第 i 个蝙蝠在 t 时刻和 $t-1$ 时刻的速度。X_{Gbest} 是通过所有蝙蝠搜索到的解中结果最好的蝙蝠个体所在的位置;X_i^t 和 X_i^{t-1} 分别是第 i 个蝙蝠在 t 时刻和 $t-1$ 时刻所在的位置。

对于局部搜索部分,选择最优解蝙蝠按照随机游走法则生成一只新的蝙蝠从而得到局部新解:

$$X_{\text{new}} = X_{\text{Gbest}} + \varepsilon \times A^t, \tag{4}$$

其中 ε 是[-1,1]的一个随机变量,A^t 是 t 时刻中所有蝙蝠的平均响度。

此外,响度 A_i 和脉冲发射率 r_i 也随着迭代的进行而相应地更新。响度和脉冲的更新公式分别为

$$A_i^t = a \times A_i^{t-1}, \tag{5}$$

$$r_i^t = r_i^0 \times [1 - \exp(-yt)], \tag{6}$$

其中 α 和 y 取常数,对于任何 $0<a<1$ 和 $0<y$,当 $t \to \infty$ 有

$$A_i^t \to 0, r_i^t \to r_i^0, \tag{7}$$

A_i^t 是蝙蝠 i 在 t 时刻发出的响度,r_i^0 和 r_i^t 分别是蝙蝠 i 在初始时刻和 t 时刻发出的脉冲。

1.2 萤火虫算法

萤火虫算法是 Yang[3]在 2008 年提出的,他把空间各点看成萤火虫,利用发光强的萤火虫会吸引发光弱的萤火虫的特点来进行寻优。在发光弱的萤火虫向发光强的萤火虫移动的过程中,完成位置的迭代,从而找出最优位置,即完成寻优过程[12]。

萤火虫的相对亮度:

$$I = I_0 e^{-yr_{ij}}, \tag{8}$$

其中,I_0 表示最亮萤火虫的亮度,即自身($r=0$ 处)荧光亮度,与目标函数值相关,即目标函数值越好,自身亮度越高;y 表示光吸收系数,因为荧光会随着距离的增加和传播媒介的吸收逐渐减弱,所以设置光强吸收系数以体现此特性,可设置为常数;r_{ij} 表示萤火虫 i 与 j 之间的距离。

萤火虫的相互吸引度 β:

$$\beta(r) = \beta_0 e^{-yr_{ij}^2}, \tag{9}$$

其中,β_0 表示最大吸引度,即光源处($r=r_0=0$ 处)的吸引度。

位移更新公式:

$$X_i^t = X_i^{t-1} + \beta[X_j^{t-1} - X_i^{t-1} + a(rand - \frac{1}{2})], \tag{10}$$

其中 X_i^t 与 X_j^t 表示 i,j 两个萤火虫在 t 时刻的位置,α 为步长因子,$rand$ 为[0,1]上服从均匀分布的随机数。

1.3 混合算法

根据萤火虫的位置更新公式(10)可知,当两只萤火虫距离靠近时将会进一步加速收敛导致陷入早熟,大量萤火虫聚集在局部最优从而错过全局最优解。本文对萤火虫算法的局部搜索进行改进,将蝙蝠算法中最优解随机游走的思想融入到萤火虫位置更新过程中,同时对该游走进行高斯扰动,增加种群的多样性,使得种群的局部搜索能力更强,并避免过快地陷入局部最优。我们将两种算法融合在一起的算法称

为蝙蝠萤火虫混合算法(Hybrid bat and firefly algorithm,HBAFA)。

首先初始化蝙蝠种群 X_i,在全局最优的个体 X_{Gbest} 附近进行局部寻优,产生的新个体 $BatX$,即当第 i 个蝙蝠 X_i 发出的脉冲 r_i 小于产生的随机生成的脉冲,则对全局最优个体 X_{Gbest} 进行高斯扰动产生一个新的个体并存放在 $BatX$ 中。如果 $BatX$ 的适应度值优于 X_i 且其响度 A_i 大于随机生成的响度,则用 $BatX$ 替换 X_i。其高斯扰动的公式如下:

$$BatX = X_{Gbest} \times (N(0,1) + 1) \quad \text{as} \quad r_i < rand, \tag{11}$$

其中 X_{Gbest} 是全局最优个体,N(0,1)是服从均值为0,方差是1的高斯分布函数,r_i 是第 i 个蝙蝠个体发出的脉冲,$rand$ 是在[0,1]上服从均匀分布的随机数。

$$X_i = BatX \text{ as } f(BatX) < f(X_i) \&\& A_i > rand, \tag{12}$$

其中 X_i 是个体 i 的位置,A_i 是个体 i 发出的响度,$rand$ 是在[0,1]上服从均匀分布的随机数。

蝙蝠通过在 X_{Gbest} 周围发射脉冲信号产生局部解 $BatX$ 来实现局部最优搜索,并通过该蝙蝠个体发出的响度和其位置的优劣来更新种群,同时通过萤火虫算法进行全局最优搜索,与萤火虫算法实现了信息上的交互,增加了萤火虫的种群多样性,加强了算法的局部搜索能力,弥补了萤火虫算法在局部搜索上的不足,使萤火虫算法得到了优化。

1.4 HBAFA 算法流程

通过上述 HBAFA 思想分析,下面给出 HBAFA 算法的详细优化步骤。

HBAFA 算法步骤如下:

Step 1 初始化相关参数,随机生成蝙蝠种群,并根据公式(11)生成 $BatX$;

Step 2 若迭代次数未到达设定次数,转 Step 3,否则结束算法;

Step 3 判断个体的脉冲 r_i 是否小于随机数 $rand$,小于则生成 $BatX$,否则转 Step 4;

Step 4 若 $f(BatX) < f(X_i) \&\& A_i > rand$,则使用公式(12)将萤火虫个体替换成蝙蝠个体,并且使用公式(5)和(6)分别更新响度 A_i 和脉冲 r_i,否则转 Step 5;

Step 5 计算任意两只萤火虫之间的欧式距离和相互吸引度 β,然后使用公式(10)更新萤火虫个体的位置;

Step 6 判断是否达到终止条件(一般为达到最大迭代次数或者达到设定精度),若达到终止条件,则迭代终止,否则转向 Step 2。

2 结果与分析

2.1 测试函数

选择6个常用的测试函数来检验混合算法的有效性,测试函数如表1所示。

表1 测试函数

Table 1 Test function

Fun Name	Fun	Range	Best
Sphere	$f_1(x) = \sum_{i=1}^{n} X_i^2$	[−100,100]	0
Rosenbrock	$f_2(x) = \sum_{i=1}^{n}[100(x_{i+1} - x_i)]^2 + (x_i - 1)^2$	[−10,10]	0
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\frac{x_i}{\sqrt{i}} + 1$	[−600,600]	0
Rastrigin	$f_4(x) = \sum_{i=1}^{n}(x^2 - 10\cos(2\pi x_i) + 10)$	[−5.12,5.12]	0
Ackley	$f_5(x) = -20 \times e^{-0.2\times\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$	[−32,32]	0
Schaffer's F6	$f_6(x) = \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} + 0.5$	[−100,100]	0

2.2 实验环境和参数设置

算法测试操作系统为 Windows 7,CPU 为 Inter E5-1620v3,主频 3.5 Ghz,内存 16 GB,Matlab2016a。设置种群大小 Popsize 均为 40,寻优精度设置为 10^(−5),最大迭代次数设置为 1 000 次。HBAFA 初始音量 $A_0 = 1$,最小音量 $A_{min} = 0$;最大脉冲值 $r_{max} = 1$,最小脉冲值 $r_{min} = 0$;萤火虫的步长因子 $\alpha = 0.2$;光吸收系数 $y = 1$。BA 初始音量也设置为 $A_0 = 1$,最小音量 $A_{min} = 0$,最大脉冲值 $r_{max} = 1$,最小脉冲值 $r_{min} = 0$,α 和 y 取 0.9。SPSO 粒子移动速度上下限设置为搜索空间的 10%。萤火虫算法步长因子 $\alpha = 0.2$,光吸收系数 $y = 1$。

2.3 收敛性分析

为降低偶然因素对实验结果的影响,我们对表1的测试函数进行 30 次独立重复实验,并记录每种算

法在每次迭代结束后所得到的实验数据,在 30 次实验中所得到的最差值 Worst、最优值 Best、平均值 Avg 以及标准偏差 Std 如表 2 所示。

表 2　测试函数实验比较

Table 2　Comparison of test function experiments

Fun	Dim	Alg	Best	Worst	Avg	Std
f_1	30	HBAFA	0	0	0	0
		BA	3.716E+3	1.668E+4	1.054E+4	3.182E+3
		FA	5.126E+4	7.700E+4	6.539 E+4	6.295E+3
		SPSO	3.442E+0	4.343E+1	2.160E+1	1.015E+1
f_2	30	HBAFA	3.232E−5	2.871E+1	2.488E+1	9.923E+0
		BA	2.614E+7	1.332E+9	5.877E+8	2.877E+8
		FA	1.408E+10	3.814E+10	2.909E+10	5.692E+9
		SPSO	6.566E+3	6.744E+5	4.705E+4	1.200E+5
f_3	30	HBAFA	0	0	0	0
		BA	1.869E+0	4.736E+0	3.611E+0	0.683E+0
		FA	1.455E+1	2.099E+1	1.763E+1	1.681E+0
		SPSO	0.353E+0	0.840E+0	0.587E+0	0.129E+0
f_4	30	HBAFA	0	0	0	0
		BA	1.417E+4	2.239E+4	1.936E+4	2.047E+3
		FA	4.900E+4	7.670E+4	6.402E+4	6.274E+3
		SPSO	1.844E+2	4.662E+2	3.321E+2	6.983E+1
f_5	30	HBAFA	8.882E−16	8.882E−16	8.882E−16	0
		BA	2.000E+1	2.000E+1	2.000E+1	3.906E−4
		FA	2.150E+1	2.150E+1	2.136E+1	0.077E+0
		SPSO	6.022E+0	2.047E+1	1.883E+1	4.147E+0
f_6	2	HBAFA	0	0	0	0
		BA	0.010E+0	0.466E+0	0.234E+0	0.140E+0
		FA	0.010E+0	0.415E+0	0.172E+0	0.120E+0
		SPSO	0	0.010E+0	0.003E+0	0.004E+0

f_1函数是二次函数型,在球面上具有单峰特性,也就是最值唯一极值。由图 1 可知,在优化f_1函数的过程中,HBAFA 的收敛曲线十分陡峭,说明其收敛速度极快,较短的迭代次数就能达到非常高的寻优精度;而 FA、SPSO 和 BA 的收敛曲线较平缓,说明这 3 种算法的种群多样性较差,极易陷入局部最优解,收敛速度明显慢于 HBAFA 算法。在寻优过程中,HBAFA 不到 100 次迭代,就能寻找到该函数的最优解 0(由于寻优曲线图的纵坐标是 $\log_{10}$(Fitness),当寻找到最优解时寻优曲线在图上无法显示);而其他 3 种算法的寻优能力较差,在迭代终止时也没能寻找到最优解。由表 2 中 f_1 的实验数据可知,在迭代结束(Iteration=1 000)时,HBAFA 寻找到的平均解优于其他 3 种算法,且方差为 0,说明该算法表现十分稳定,每次都能够在迭代结束前寻找到 Sphere 函数的最优解。综上所述,在 Sphere 函数的寻优过程中,HBAFA 的收敛速度、收敛精度和稳定性都优于其他 3 种算法。

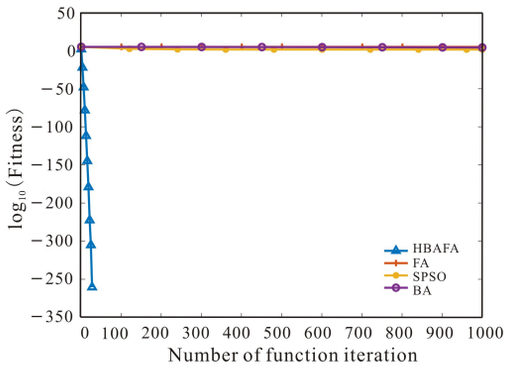

图 1　f_1函数对比曲线

Fig. 1　Contrast curve of function f_1

f_2与其他 5 个函数图像不同,该函数在定义域内具有连续性,只有一个峰值,对应全局最小却在局部最优之间的波谷下,检测过程中极易检测到局部最优,但很难找到全局最佳点,是算法开发与尝试方面极其经典的检测函数。由图 2 可知,在优化 f_2 函数的前期,在迭代次数相同时,HBAFA 的收敛速度和收敛精度均优于其他 3 种算法;在寻优后期,4 种算法均陷入局部最优解;在迭代结束之后,4 种算法均没有找到最优解。由表 2 中 f_2 的实验数据可知,在迭代结束之后,HBAFA 寻找到的平均解 Avg 优于其他 3 种算法,因此 HBAFA 的收敛精度均优于其他 3 种算法。综上所述,HBAFA 整体的收敛性能均优于其他 3 种算法。

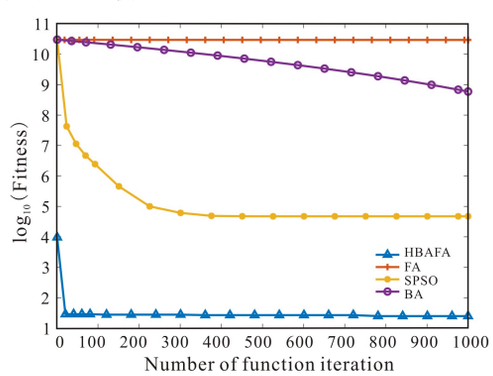

图 2　f_2函数对比曲线

Fig. 2　Contrast curve of function f_2

f_3是高次函数型,其图像为非线性变化,都具有多个极值和多个图像峰,这些极值服从某种函数分布,可检验算法在多模以及高维的收敛情况。由图 3 可知,在优化 f_3 函数的前期,在迭代次数相同时,HBAFA 的收敛速度和收敛精度均优于其他 3 种算法;在寻优后期,BA、SPSO 和 FA 这 3 种算法均陷入局部最优解;在迭代结束之后,只有 HBAFA 找到最优解,并且在迭代次数不到 100 次就能寻找到最优

解。由表 2 中f_3的实验数据可知,在迭代结束之后,HBAFA 寻找到的平均解 Avg 优于其他 3 种算法,并且寻找到最优解,因此 HBAFA 的收敛精度均优于其他 3 种算法。综上所述,HBAFA 整体的收敛性能均优于其他 3 种算法。

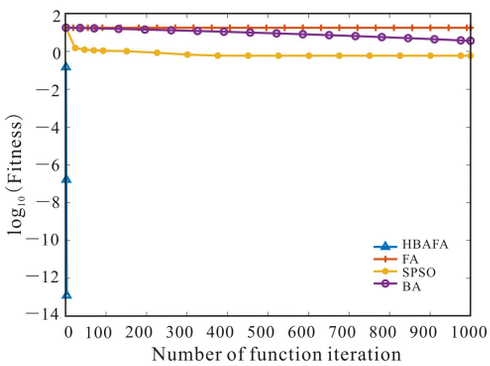

图 3　f_3函数对比曲线

Fig. 3　Contrast curve of function f_3

f_4从函数分析、最值求解及其图像中不难看出,该函数存在大概 10^n 个局部极小值,其中 n 表示空间的维数。由图 4 可知,在优化f_4函数的前期,在迭代次数相同时,HBAFA 的收敛速度和收敛精度均优于其他 3 种算法;在寻优后期,BA、SPSO 和 FA 这 3 种算法均陷入局部最优解;在迭代结束之后,只有 HBAFA 找到最优解,并且在迭代次数不到 100 次就能寻找到最优解。由表 2 中f_4的实验数据可知,在迭代结束之后,HBAFA 寻找到的平均解 Avg 优于其他 3 种算法,并且寻找到最优解,因此 HBAFA 的收敛精度均优于其他 3 种算法。综上所述,HBAFA 整体的收敛性能均优于其他 3 种算法。

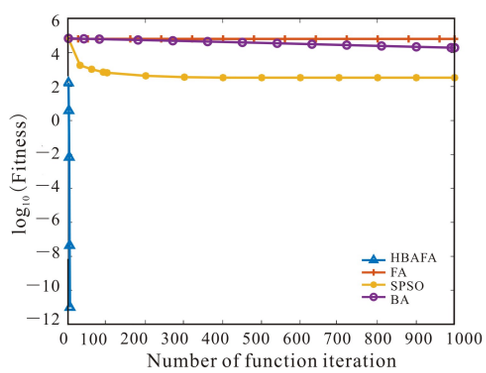

图 4　f_4函数对比曲线

Fig. 4　Contrast curve of function f_4

f_5的函数图像复杂多变,所对应的搜索要求特别不易控制,检测过程中容易出现局部极值问题。由图 5 能够看出,在优化 Ackley 函数的过程中,HBAFA 在初期十分陡峭,说明在搜索初期收敛速度极快,较短的迭代次数就能达到非常高的寻优精度;而 FA、SPSO 和 BA 的收敛曲线较平缓,说明这 3 种算法的种群多样性较差,极易陷入局部最优解,收敛速度明显慢于 HBAFA 算法,且收敛精度低,远不能达到设定的收敛精度要求。由表 2 中f_5的实验数据可知,在迭代结束时,HBAFA 寻找到的平均解优于其他 3 种算法,说明该算法表现十分稳定,每次都能够在迭代结束前寻找到 Ackley 函数较高精度的解。综上所述,在 Ackley 函数的寻优过程中,HBAFA 的收敛性能都优于其他 3 种算法。

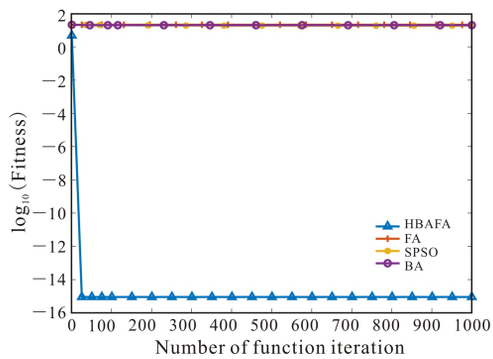

图 5　f_5函数对比曲线

Fig. 5　Contrast curve of function f_5

f_6在检测过程中极不稳定,振荡现象非常明显,幅值很大。由图 6 能够看出,在优化 Schaffer's F6 函数的过程中,HBAFA 的收敛曲线在初期非常陡峭,说明在搜索初期收敛速度极快,不到 100 次迭代就能够搜索到 Schaffer's F6 函数的最优解 0;而 FA 和 BA 的收敛曲线则十分平缓,两者收敛速度相当,收敛精度差别也不大,SPSO 则有较陡峭的收敛曲线,且收敛精度略优于 FA 和 BA,但都无法达到搜索精度的要求。由表 2 中f_6的实验数据可知,在迭代结束(Iteration=1 000)前,HBAFA 寻找到的平均解优于其他 3 种算法,说明该算法表现十分稳定,每次都能够

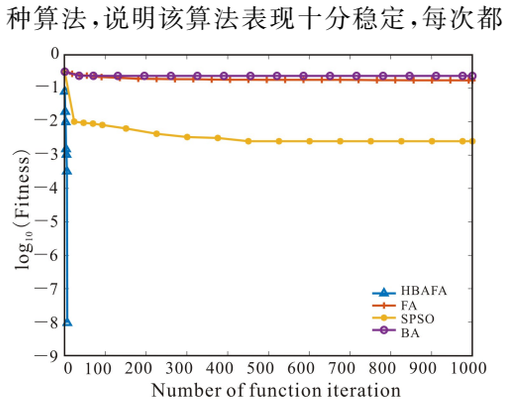

图 6　f_6函数对比曲线

Fig. 6　Contrast curve of function f_6

在迭代结束前寻找到 Schaffer's F6 函数的最优解。综上所述,在 Schaffer's F6 函数的寻优过程中,HBAFA 的收敛性能都优于其他 3 种算法。

HBAFA,FA,SPSO,BA 相对应的检测显示输出曲线如图 1～6 所示,FA 和 BA 的表现相差不大,SPSO 在函数 f_2,f_3,f_4,f_6 的性能比这两者优越,但效果也并不是十分理想。而 HBAFA 无论是在单峰函数 Sphere、Rosenbrock 还是多峰函数 Griewank、Rastrigin、Ackley 以及 Schaffer,都能在迭代初期就有很好的表现,输出的适应度值好,收敛速度快,尤其是 f_1,f_3,f_4,f_6 在很早期就能够达到收敛。f_3,f_4,f_5 具有多个局部极值,可检验算法在多模以及高维的收敛情况,FA、SPSO、BA 对这 3 个函数均无法寻找到最佳解,而 HBAFA 对于 f_3、f_4 能够寻到最优解,而对 f_5 能够找到满足精度要求的解。函数 f_2 的全局最优在局部最优之间的谷底,检测过程中极易陷入局部最优,FA、SPSO、BA 均无最佳解,而 HBAFA 能够寻到满足精度要求的解。综上数据分析可知,HBAFA 改良了 BA 和 FA,算法优于 BA,FA 和 SPSO,具有更高的寻优精度,更高的稳定性,说明该算法具有一定的优越性。

2.4 鲁棒性分析

标准偏差 Std 的大小反映了算法鲁棒性的好坏,即标准偏差越小,则算法的鲁棒性越好;反之,则算法的鲁棒性越差。由表 2 的实验数据可知,在 f_1～f_6 的寻优过程中,HBAFA 的标准偏差 Std 均小于其他 3 种算法的标准偏差,因此 HBAFA 总体的鲁棒性均优于其他 3 种算法。

2.5 成功率比较

设定参考寻优精度值,在每一次迭代结束之后,判断适应度值是否达到目标精度 10^(−5),若达到则停止迭代,并记录一次寻优成功;若迭代次数为 1 000 次时仍未达到目标精度,则终止迭代。在 30 次独立重复试验中,4 种算法的寻优成功率如表 3 所示。由表 3 可知,将寻优精度设置为 10^(−5),则在迭代结束后,4 种算法在 f_2 寻优成功率均为 0%,而 HBAFA 在剩余其他 5 个基准函数的寻优成功率均为 100%;BA、SPSO 和 FA 的整体寻优成功率稍差。因此,在 10 个基准函数的寻优过程中,HBAFA 总体的寻优成功率优于其他 3 种算法。

2.6 算法优越性分析

HBAFA 的整体性能优于其他 3 种算法,主要源自以下 3 个方面:

表 3 成功率比较(%)

Table 3 Comparing with success rate (%)

Fun	HBAFA	FA	BA	SPSO
f_1	100	0	0	0
f_2	0	0	0	0
f_3	100	0	0	0
f_4	100	0	0	0
f_5	100	0	0	0
f_6	100	0	0	73.3

第一,由于蝙蝠算法具有较强的局部搜索能力,蝙蝠种群根据蝙蝠发出的脉冲和响度在全局最优个体 Gbest 在附近进行了更加详细的局部搜索,并提高了混合算法的收敛性能;

第二,将蝙蝠算法引入到萤火虫算法,协助萤火虫种群寻优。在寻优过程中加强了两个种群之间的信息交流,以提高种混合算法的收敛性能;

第三,在寻优过程中,对全局最优个体 Gbest 进行高斯扰动,提高了种群的多样性,避免种群过早地陷入局部最优。

3 结论

本文通过将蝙蝠算法的局部搜索机制引入到萤火虫算法中,让蝙蝠种群协助萤火虫种群寻优;两个种群在寻优过程中加强信息交互,以提高混合算法的收敛性能;而对全局最优个体进行高斯扰动,增加了种群的多样性,避免种群过快陷入局部最优解。从萤火虫算法、标准粒子群算法、蝙蝠算法和本文提出的蝙蝠萤火虫混合算法作仿真对比实验中,可以得出这样的结论:相比其他 3 种算法,蝙蝠萤火虫混合算法具有相对更快的收敛速度,更高的收敛精度以及更好的鲁棒性。在未来的工作中,我们将对蝙蝠烟花混合算法进行更深入的理论研究,并将其运用到更多的具体工程实践中。

参考文献

[1] KENNEDY J,EBERHART R. Particle swarm optimization [C]// Proceedings of ICNN'95-International conference on Neural Networks, Ⅳ. Piscataway, NJ: IEEE Service Center,1995:1942-1948.

[2] YANG X S. A new metaheuristic bat-inspired algorithm [M]//Nature inspired cooperative strategies for optimization. Berlin:Springer,2010:65-74.

[3] YANG X S. Nature-inspired metaheuristic algorithm

[M]. Beckington, UK: Luniver Press, 2010.
[4] 王艳,王秋萍,王晓峰. 基于改进萤火虫算法求解旅行商问题[J]. 计算机系统应用,2018,27(8):219-225.
[5] ŁUCKASIK S, ŻAK S. Firefly algorithm for continuous constrained optimization task [C]//ICCCI 2009: Proceedings of the First International Conference on computational collective intelligence, semantic web, social networks and multi-agent systems, LNCS5796. Berlin: Springer, 2009: 97-100.
[6] 冯艳红,刘建芹,贺毅朝. 基于混沌理论的动态种群萤火虫算法[J]. 计算机应用, 2013,33(3):796-799,805.
[7] 王吉权,王福林. 萤火虫算法的改进分析及应用[J]. 计算机应用,2014,34(9):2552-2556.
[8] GANDOMI A H, YAN X S, TALATAHARI A H, et al. Firefly algorithm with chaos [J]. Communications in Nonlinear Science and Numerical Simulation, 2013, 18(1): 89-98.
[9] 付强,葛洪伟,苏树智. 引入萤火虫行为和 Levy 飞行的粒子群优化算法[J]. 计算机应用,2016,36(12):3298-3302,3310.
[10] 郝晓莹,贺兴时,薛菁菁. 一种粒子群-萤火虫算法的参数优化方法[J]. 西安工程大学学报,2017,31(5):695-700.
[11] 杜艳艳,刘升. 一种改进的自适应混合型蝙蝠算法[J]. 微电子学与计算机,2018,35(6):135-140.
[12] 左仲亮,郭星,李炜. 一种改进的萤火虫算法[J]. 微电子学与计算机,2018,35(2):61-66.

The Hybrid Bat and Firefly Algorithm with Collaborative Optimization

WEN Tai, ZHAO Zhigang, MO Haimiao
(College of Computer and Electronics Information, Guangxi University, Nanning, Guangxi, 530004, China)

Abstract: The improved firefly algorithm has defects such as dependence on the initial solution distribution, slow convergence in the later stage, easy stagnation, early maturity and low accuracy. This algorithm introduces the bat population to do a more detailed local search near the global optimum to help the firefly population to optimize. In the process of optimization, the information interaction between the bat population and the firefly population is enhanced, and the optimization is coordinated. Finally, Gaussian perturbation is applied to the global optimum individuals to increase the diversity of the population and avoid the population falling into the local optimum solution. The algorithm was tested by using six benchmark functions and compared with the other three algorithms (standard particle swarm algorithm, bat algorithm, firefly algorithm). The results show the overall performance of this hybrid algorithm is better than the other three algorithms. The introduction of the bat population has greatly improved the performance of firefly, and the improvement is effective.

Key words: function optimization, bat algorithm, firefly algorithm, collaborative optimization

责任编辑:符支宏