DOI:10.11721/cqnuj20170401

基于动态分组与高斯扰动的改进布谷鸟算法*

薛益鸽[1], 邓辉文[2]
(1. 温州商学院 信息工程学院, 浙江 温州 325035; 2. 西南大学 计算机与信息科学学院, 重庆 400715)

摘要:【目的】针对原有布谷鸟算法在求解最优化问题时的不足,提出一种基于动态分组与高斯扰动的改进布谷鸟搜索算法(Gaussian perturbating and dynamic grouping for cuckoo search,GPDGCS)。【方法】GPDGCS 算法在原有布谷鸟算法的求解过程中应用了高斯扰动与动态分组策略。【结果】通过 6 个典型的测试函数对 GPDGCS 算法进行仿真实验,结果表明 GPDGCS 算法比原有布谷鸟算法有更高的收敛速度、求解精度等。【结论】GPDGCS 算法在一定程度上可避免算法陷入局部最优。

关键词:布谷鸟搜索算法;高斯扰动;动态分组
中图分类号:TP301.6　　**文献标志码**:A　　**文章编号**:1672-6693(2018)02-0108-06

20 世纪以来,出现了各种智能优化算法,如遗传算法(Genetic algorithm,GA)[1]、粒子群算法(Particle swarm optimizaton,PSO)[2]等;近几年还出现了一系列启发式智能优化算法,如蝙蝠算法(Bat algorithm,BA)[3]、布谷鸟算法(Cuckoo search,CS)[4]等。其中布谷鸟算法是 Yang[4]根据布谷鸟寻窝产卵的行为而提出的一种新颖的启发式智能优化算法。

布谷鸟算法自提出之日起便得到国内外学者的强烈关注,他们对布谷鸟算法进行了研究和改进。贺兴时等人[5]针对原有布谷鸟算法容易陷入局部收敛的问题,将原有布谷鸟算法与高斯扰动结合,提出一种基于高斯扰动的布谷鸟搜索算法,这一算法与原有布谷鸟算法相比能更好地跳出局部收敛。针对原有布谷鸟搜索算法在寻求全局最优解问题上存在收敛精度不够高的问题,Valian 等人[6]提出基于全局优化问题的布谷鸟算法,此算法能较快速地在可行域内求解全局最优解。针对原有布谷鸟算法求解全局最优解速度不够快的问题,尹义龙等人[7]提出一种逐维改进的布谷鸟搜索算法,该算法把对原有布谷鸟算法的改进从全局变成了逐维改进,即将改进方法从整体改进变成了对解的每个维度进行求解最优,最后通过仿真实验验证了算法改进的有效性。王李进等人[8]提出可变步长控制量的布谷鸟算法,通过自适应地调整布谷鸟算法的步长控制量在很大程度上提高了布谷鸟算法的求解速度。Xue 等人[9]利用步长控制向量在布谷鸟算法寻优过程中的作用,使用动态分组的方式对布谷鸟算法进行改进,并得到 DGCS(Dynamic grouping for cuckoo search),这一算法能在很大程度上解决布谷鸟算法求解速度不够快、求解精度不够高的问题。基于上述研究,本研究将对原有布谷鸟算法在动态分组的基础上再加入高斯扰动进行改进。

1 布谷鸟算法

布谷鸟采用寄生育雏的繁殖策略,即布谷鸟将自己的鸟蛋产于其他鸟类的鸟巢之中。但是鸟巢的主人有一定的概率能发现外来鸟蛋。如果鸟巢的主人发现外来鸟蛋,就会将外来鸟蛋毁坏或者放弃自己的鸟巢。在此基础上,Yang 给出布谷鸟算法。此算法对布谷鸟寄生育雏的活动进行模拟,且满足以下 3 个假设[4,6,10]:

1) 每只布谷鸟每次产 1 个布谷鸟蛋,且随机选择鸟巢进行存放;

2) 每次进化都保留最优的存有布谷鸟蛋的鸟巢到下一代;

3) 存有布谷鸟蛋的鸟巢数量不变,布谷鸟蛋被鸟巢主人发现的概率为一固定值 $p_a \in [0,1]$。

基于以上 3 个假设,CS 算法的步骤可以总结如下:

* **收稿日期**:2017-05-10　**修回日期**:2018-01-17　**网络出版时间**:2017-05-16 11:28
资助项目:重庆社科规划项目(No.2013YBZX008);重庆市文科研究基地重点项目(No.16SKB040);国家社会科学基金重大项目(No.14ZDB016);浙江省教育厅科研项目(No.Y201636359)
第一作者简介:薛益鸽,男,助教,研究方向为计算智能,E-mail:15968723096@163.com
网络出版地址:http://kns.cnki.net/kcms/detail/50.1165.N.20170516.1128.102.html

1）初始化。设算法种群的规模为 n，随机生成 n 个鸟巢位置，计算这些鸟巢位置的适应值，最优鸟巢位置记为 $x_{0,b}$，$b\in\{1,2,\cdots,n\}$，最优适应值记为 $f_{\min}$，最大进化次数为 K，解的搜索空间维数 m，布谷鸟蛋被鸟巢主人发现的概率为 p_a；

2）循环体。保留上次进化中种群适应值最优的鸟巢位置，记为 $x_{k-1,b}$；k 为正整数，表示当前的进化代数，使用(6)式对当前鸟巢位置 $l_{k-1}=[x_{k-1,1},x_{k-1,2},\cdots,x_{k-1,n}]$ 进行 Lévy-flights 随机游动更新，并比较 l_{k-1} 与 l_{k-1} 更新后解的适应值，保留适应值较优的解，记为 $l_k=[x_{k,1},x_{k,2},\cdots,x_{k,n}]$；

3）对 l_k 中每个解都生成一个随机数 $r\sim U(0,1)$，r 为外来布谷鸟蛋被发现的概率。若 $r>p_a$，则按(7)式对该鸟巢位置进行更新，比较更新前和更新后的鸟巢位置的适应值，保留较优的鸟巢位置，记为 $j_k=[x_{k,1},x_{k,2},\cdots,x_{k,n}]$；否则保留该鸟巢的位置不变；

4）求出 j_k 中的最优鸟巢位置 $x_{k,b}$ 并算出此解的适应值 $f_{\min}$，判断此时是否满足算法终止条件。满足算法终止条件则输出 $f_{\min}$ 以及相对应的全局最优解 $x_{k,b}$；否则，执行步骤 2，继续进行进化操作，直到算法终止条件满足。

上述算法中，采用 Lévy-flights 随机游动生成新解 $x_{k+1,i}$ 的过程，就是执行(1)式的操作：

$$x_{k+1,i}=x_{k,i}+\alpha\oplus \text{Lévy}(\lambda)。\tag{1}$$

其中，$x_{k,i}$ 表示第 k 代第 i 个解；α 则是步长控制向量，步长控制向量用于控制搜索步长的大小，也可以说是控制算法搜索的范围，其维数和布谷鸟算法中种群规模相等。文献[11]中使用(2)式来计算步长控制向量，这样可以从当前代最优解中得到更多有用的信息。

$$\alpha=\alpha_0(x_{k,i}-x_{k,\text{best}})。\tag{2}$$

其中，$\alpha_0=0.01$，$x_{k,\text{best}}$ 表示第 k 代的最优解。

(1)式中，$\oplus$ 表示点乘积，Lévy(λ)为 Lévy 随机搜索路径，它的移动步长服从 Lévy 概率分布：

$$\text{Lévy}(\lambda)\sim u=t^{-1-\lambda},0<\lambda\leqslant 2。\tag{3}$$

为了便于计算，文献[11]采用(4)式计算 Lévy 随机数：

$$\text{Lévy}(\lambda)\sim\frac{\varphi\times u}{|v|^{\frac{1}{\lambda}}},\tag{4}$$

其中，u，v 服从正态分布，即 $u\sim N(0,1)$，$v\sim N(0,1)$，$\lambda=1.5$ 及

$$\varphi=\left(\frac{\Gamma(1+\lambda)\times\sin\left(\pi\times\frac{\lambda}{2}\right)}{\Gamma\left(\left(\frac{1+\lambda}{2}\right)\times\lambda\times 2^{\frac{(\lambda-1)}{2}}\right)}\right)^{\frac{1}{\lambda}};\tag{5}$$

这里 Γ 为标准的 Gamma 函数。

综合(1)～(5)式，在 CS 算法的 Lévy-flights 随机游动组件中，采用(6)式生成新的解 $x_{k+1,i}$，$i\in\{1,\cdots,n\}$：

$$x_{k+1,i}=x_{k,i}+\alpha_0\frac{\varphi\times u}{|v|^{\frac{1}{\lambda}}}(x_{k,i}-x_{k,\text{best}})。\tag{6}$$

在按发现概率丢弃部分解之后，使用(7)式重新生成数量的新解：

$$x_{k+1,i}=x_{k,i}+r(x_{k,j}-x_{k,e})。\tag{7}$$

其中，r 为缩放因子，是(0,1)区间内服从均匀分布的随机数；$x_{k,j}$ 和 $x_{k,e}$ 是表示第 k 代的两个随机解。

2 基于动态分组与高斯扰动的改进布谷鸟搜索算法

2.1 基于高斯变异算法的算法改进策略

智能优化算法，包括布谷鸟算法的寻优过程都依赖于个体间的相互作用。算法寻优过程中一旦陷入局部最优就会很大程度上影响求解精度和收敛速度；在寻优过程中，解集缓慢地向当前全局最优解靠近，这样就降低了种群多样性。布谷鸟算法中引入高斯扰动可以一定程度上避免布谷鸟算法在寻优过程中陷入局部最优，可以提高种群多样性。为了判断算法在寻优过程是否陷入局部最优，在求解过程中对布谷鸟算法解集的当前代最优解的适应值设立公告板进行记录，如果连续 3 代最优解的适应值没有改变或者改变非常微小，则判定为陷入局部最优，此时用当前代最优解的位置替代当前代最差解，并对解集中除最优解和最差解之外的中间解集进行高斯扰动而得到新的解集，将新解集的适应值与公告板中解的适应值进行比较，若新解集中存在解的适应值优于公

告板中解的适应值,则修改公告板中的解,并将新解集留到下一代进行寻优。

高斯扰动是在原有解的基础上加一个服从高斯变异的随机扰动量,公式为:

$$x'_{k,i}=x_{k,i}+x_{k,i}\times(2\times N(0,1)-1), \tag{8}$$

其中,$N(0,1)$服从标准高斯分布。

2.2 基于步长控制向量的动态分组策略

设布谷鸟的种群规模为 n,种群中第 k 代第 i 个布谷鸟位置 $x_{k,i}$ 的适应值为 f_i,种群中的最优位置的适应值为 f_{best};种群中所有鸟巢位置的平均适应值为 $f_a=(f_1+f_2+\cdots+f_n)/n$,对种群中的适应值优于 f_a 的个体求平均适应值,记为 f'_a。根据 f_a,f'_a 将种群分为3组。步长控制向量的调整方案如下:

1) 将鸟巢位置适应值优于 f'_a 的鸟巢位置归为第一组,并对步长控制向量的对应分量做以下调整:

$$\alpha_{k+1,i}=\alpha_{k,i}-(\alpha_{k,i}-\alpha_{\min})\left|\frac{f_i-f'_a}{f_{\text{best}}-f'_a}\right|。 \tag{9}$$

2) 将鸟巢位置适应值次于 f_a 的鸟巢位置归为第二组,并对步长控制向量的对应分量做以下调整:

$$\alpha_{k,i}=1-\frac{0.8}{1+h_1\exp(-h_2\Delta)}。 \tag{10}$$

此处 h_1,h_2 和 Δ 均为控制参数,$\Delta=|f_{\text{best}}-f'_a|$。

3) 将鸟巢位置适应值介于 f_a 与 f'_a 之间的鸟巢位置归为第三组,对步长控制向量的对应分量不做改变。

2.3 改进算法的流程

1) 初始化。设种群规模为 n,最大寻优代数为 K,寻优空间维数为 m,初始步长控制向量 α 中每个分量的值均为 β,最小高斯偏移量 q,连续适应值改变不明显代数 a,种群中的最优鸟巢位置记为 $x_{0,b}$,其中 $b\in\{1,2,\cdots,n\}$ 对应的最优适应值为 $f_{\min}$,并在公告板中记录 $x_{0,b}$ 与 $f_{\min}$;

2) 循环体。使用(9)、(10)式更新种群的步长控制向量,然后使用(6)式对上一代的解集 $j_{k-1}=[x_{k-1,1},x_{k-1,2},\cdots,x_{k-1,n}]$ 进行更新得到新的鸟巢位置,比较新的鸟巢位置与 $j_{k-1}=[x_{k-1,1},x_{k-1,2},\cdots,x_{k-1,n}]$ 的适应值,保留适应值较优的鸟巢位置到步骤3,记为 $l_k=[x_{k,1},x_{k,2},\cdots,x_{k,n}]$。其中 k 是一个正整数,代表当前的进化代数;

3) 对 l_k 中的每个鸟巢位置分别产生一个随机数 $r\sim U(0,1)$,如果 $r<p_a$,则当前鸟巢位置不变;否则用(7)式对当前位置进行随机的更新,保留更新后的鸟巢位置与更新前鸟巢位置中适应值较优的位置进入下一代,记为 $j_k=[x_{k,1},x_{k,2},\cdots,x_{k,n}]$;

4) 找出 j_k 的当前最优位置 $x_{k,b}$ 以及对应的最优适应值 $f_{\min}$,并判断此时是否满足算法终止条件。如果满足,输出最优位置 $x_{k,b}$ 以及最优适应值;否则,执行步骤5;

5) 公告板中记录 $x_{k,b}$ 以及 $f_{\min}$,并比较当前代与公告板中最优鸟巢位置的适应值,如果当前最优鸟巢位置的适应值相对公告板中鸟巢位置的适应值变化低于 q 则让 a 自加1;判断此时 a 是否等于2,如果是则用 $x_{k,b}$ 替代 j_k 中的最差解,并用 $f_{\min}$ 替代 j_k 中的最差解的适应值,并对 j_k 中除去最优解和最差解以外的中间解集进行(8)式的高斯扰动,得到新解集 $j'_k=[x'_{k,1},x'_{k,2},\cdots,x'_{k,n}]$,判断此时 j'_k 中的最优解适应值是否优于 $x_{k,b}$,如果是则用 j'_k 替代 j_k 进入下一代的进化;否则使用 j_k 进行下一代的进化,直到算法终止条件满足。

3 实验与结果

3.1 实验设计

仿真环境:Windows 7,内存 2.00 GB,机器主频 2.90 GHz,Matlab R2016a 软件。

算法参数的设计见表1。

表1 参数设计

Tab. 1 Parameter design

算法名称	参数设计
CS	鸟巢的规模为20,$p_a=0.25$
DGCS	鸟巢的规模为20,$p_a=0.25$,$\alpha_{\min}=0.001$,$h_1=1.5$,$h_2=1$
GPDGCS	鸟巢的规模为20,$p_a=0.25$,$\alpha_{\min}=0.001$,$h_1=1.5$,$h_2=1$,$q=0.000\ 1$

本文使用如下6个基准测试函数进行测试:

1) Ackley 函数:

$$f(x)=-20\exp\left(-0.02\sqrt{D^{-1}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(D^{-1}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+\mathrm{e};$$

2）Beale 函数：

$$f(x)=(x_1x_2-x_1+1.5)^2+(x_1x_2^2-x_1+2.25)^2+(x_1x_2^3-x_1+2.625)^2。$$

3）Matyas 函数：

$$f(x)=0.26(x_1^2+x_2^2)-0.48x_1x_2。$$

4）Rosen 函数：

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

5）Sum Squares 函数：

$$f(x)=\sum_{i=1}^{D}ix_i^2。$$

6）Zacharov 函数：

$$f(x)=\sum_{i=1}^{D}x_i^2+\left(\frac{1}{2}\sum_{i=1}^{D}ix_i\right)^2+\left(\frac{1}{2}\sum_{i=1}^{D}ix_i\right)^4。$$

以上 6 个测试函数的参数见表 2。

表 2　测试函数的参数

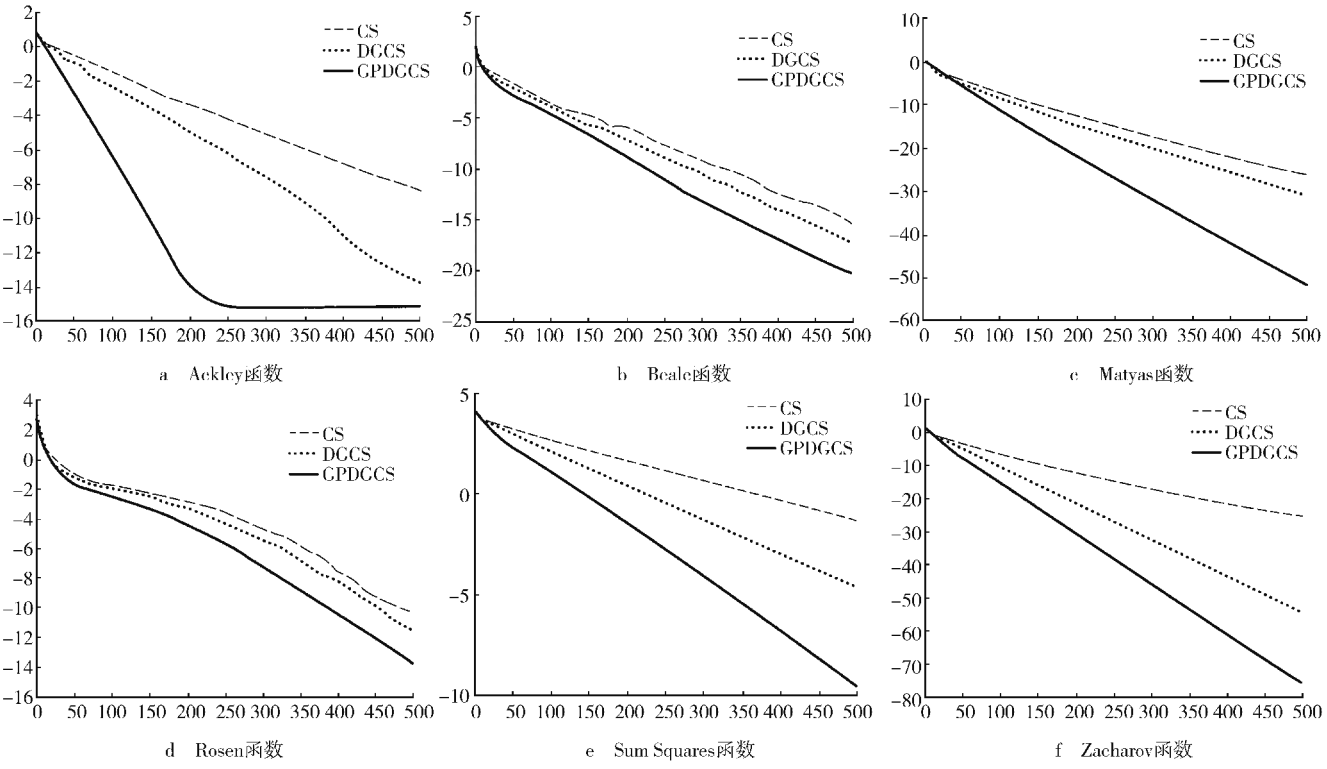
Tab. 2 The parameters of test function

函数类型	维数	搜索范围	理论最优值
Ackley	2	[−32,32]	0
Beale	2	[−10,10]	0
Matyas	2	[−10,10]	0
Rosen	2	[−5,10]	0
Sum Squares	20	[−10,10]	0
Zacharov	2	[−5,10]	0

使用上述 6 个测试函数对 GPDGCS，DGCS 和 CS 算法进行测试。取算法的寻优代数为 500，算法独立运行 30 次，记录 3 种算法每次独立运行得到的最优值，并分别对 3 种算法的 30 个最优值进行取最大值、取最小值和平均值。利用所得最大值、最小值、平均值和 30 次独立运行所得的平均收敛曲线对比图对 3 种算法进行性能比较。

3.2 实验结果分析

3.2.1 CS，DGCS 和 GPDGCS 的平均收敛曲线对比　在最大进化次数和种群规模都相同的条件下，对 CS，DGCS 和 GPDGCS 使用以上 6 个测试函数进行仿真实验，如图 1 所示。



注：以上平均收敛曲线对比图的横轴均表示寻优次数，纵轴表示 3 种算法独立运行 30 次中历史最优值的对数平均值。

图 1　各函数收敛曲线对比

Fig. 1　Comparison of the convergence curve of functions

在种群规模为20,寻优代数为500时,CS,DGCS和GPDGCS算法独立运行30次的历史最优值的对数平均值(Mean best fitness logarithm,MBFL)的变化过程由(11)式表示:

$$F_{\mathrm{MBFL}}(t)=\frac{1}{G}\sum_{i=1}^{G}\lg(g_{\mathrm{best}(i)}(t))。\tag{11}$$

其中,G 表示算法独立运行的总次数,$g_{\mathrm{best}(i)}(t)$表示算法第 i 次运行的第 t 次迭代后的历史最优值。

从以上的6张平均收敛曲线对比图看出,GPDGCS算法不管是求解精度还是收敛速度都优于DGCS算法和CS算法。

3.2.2 CS,DGCS和GPDGCS的平均迭代次数对比 在求解精度都为e-10的前提下,CS,DGCS和GPDGCS分别独立运行30次,算法每次独立运行的历史最优解的适应值低于e-10时,结束当次运行,并记录此时的算法迭代次数,算法独立运行30次后,分别取这3种算法的30次独立运行得到的30个迭代次数的平均值进行比较(表3)。

表3给出了GPDGCS,DGCS和CS算法对于以上6个测试函数在求解精度都为e-10的条件下的计算结果。在求解精度都为e-10的条件下,GPDGCS的平均寻优次数比DGCS和CS都少,GPDGCS对于Ackley,Beale,Matyas,Rosen,Sum Squares和Zacharov的平均寻优次数分别是DGCS平均寻优次数的35.4%,94.6%,65.5%,86.6%,63.6%和69.54%;GPDGCS对于Ackley,Beale,Matyas,Rosen,Sum Squares和Zacharov的平均寻优次数分别是CS平均寻优次数的23.69%,73.9%,52.9%,84.1%,37.6%和43%。可以看出,GPDGCS算法对于这6个经典测试函数的寻优结果均要优于CS算法和DGCS算法。

表3 CS,DGCS和GPDGCS的平均迭代次数对比

Tab. 3 CS, DGCS and GPDGCS average iteration number comparison

测试函数	CS	DGCS	GPDGCS
Ackley	602.100 0	402.900 0	142.666 7
Beale	346.433 3	270.433 3	256.100 0
Matyas	147.433 3	119.000 0	77.966 7
Rosen	456.333 3	443.333 3	384.000 0
Sum Squares	1 405.300 0	831.400 0	528.733 3
Zacharov	160.533 3	99.166 7	68.966 7

3.2.3 CS,DGCS和GPDGCS的全局最优值对比 为了进一步研究GPDGCS的寻优性能,在种群规模为20,进化次数为500的条件下,CS,DGCS和GPDGCS分别独立运行30次,并对这30次运行所得的最优值的最大值、最小值和平均值进行比较,如表4所示。

表4给出了CS,DGCS和GPDGCS在6个测试函数中的计算结果。从表中结果可以看出,6个经典的测试函数中,GPDGCS的计算结果均要优于CS和DGCS。

表4 CS,DGCS和GPDGCS求解函数的最小值、最大值、平均值对比

Tab. 4 CS, DGCS and GPDGCS to solve the function of the minimum, maximum, average value comparison

测试函数	算法	最小值	最大值	平均值
Ackley	CS	1.382 9e-10	7.076 8e-08	8.643 6e-09
	DGCS	8.881 8e-16	6.098 0e-11	2.683 9e-12
	GPDGCS	8.881 8e-16	8.881 8e-16	8.881 8e-16
Beale	CS	1.965 5e-20	5.442 3e-12	2.895 2e-13
	DGCS	2.460 4e-23	9.719 9e-13	5.262 7e-14
	GPDGCS	3.007 5e-30	6.021 6e-16	4.499 7e-17
Matyas	CS	7.455 0e-30	9.783 4e-24	4.611 7e-25
	DGCS	1.075 3e-35	1.009 7e-26	4.105 7e-28
	GPDGCS	1.004 1e-60	6.800 2e-47	4.130 3e-48
Rosen	CS	4.356 7e-16	1.433 0e-06	5.765 0e-08
	DGCS	9.826 7e-18	7.198 4e-09	3.928 0e-10
	GPDGCS	1.698 3e-21	6.715 4e-10	3.292 9e-11
Sum2	CS	0.014 8	0.160 6	0.056 4
	DGCS	4.173 3e-06	9.012 5e-05	3.124 0e-05
	GPDGCS	2.417 1e-12	8.619 2e-09	1.236 4e-09
Zacharov	CS	1.440 8e-30	1.967 1e-22	7.434 1e-24
	DGCS	1.965 1e-59	7.050 7e-50	2.471 2e-51
	GPDGCS	5.414 6e-83	7.436 5e-70	2.842 1e-71

4 结论

本研究提出了一种基于动态分组与高斯扰动的改进布谷鸟算法,目的是为了加快布谷鸟搜索算法的搜索速度,提高其计算精度。最后经过6个标准测试函数的测试,结果表明,改进后的布谷鸟算法提高了收敛速度、求解精度,算法更加稳定。

参考文献：

[1] GOLDBERG D E. Genetic algorithms in search, optimization and machine learning[M]. Boston: Addison-Wesley, 1989.

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

[3] YANG X S. A new metaheuristic bat-inspired algorithm[J]. Computer Knowledge & Technology, 2010, 284: 65-74.

[4] YANG X S, DEB S. Cuckoo search via Lévy flights [C]// Proceedings of World Congress on Nature & Biologically Inspired Computing. [S.l.]: IEEE, 2009: 210-214.

[5] 王凡，贺兴时，王燕. 基于高斯扰动的布谷鸟搜索算法[J]. 西安工程大学学报，2011，25(4)：566-569.
WANG F, HE X S, WANG Y. The cuckoo search algorithm based on Gaussian disturbance[J]. Journal of Xi'an Polytechnic University, 2011, 25(4): 566-569.

[6] VALIAN E, MOHANNA S, TAVAKOLI S. Improved cuckoo search algorithm for global optimization[J]. International Journal of Communications and Information Technology, 2011, 1(1): 31-44.

[7] 王李进，尹义龙，钟一文. 逐维改进的布谷鸟搜索算法[J]. 软件学报，2013(11)：2687-2698.
WANG L J, YIN Y L, ZHONG Y W. Cuckoo search algorithm with dimension by dimension improvement[J]. Journal of Software, 2013(11): 2687-2698.

[8] WANG L, YIN Y, ZHONG Y. Cuckoo search with varied scaling factor[J]. Frontiers of Computer Science, 2015, 9(4): 623-635.

[9] XUE Y G, DENG H W. The cuckoo search algorithm based on dynamic grouping to adjust flight scale[J]. Applied Mechanics and Materials, 2014, 543: 1822-1826.

[10] 薛益鸽. 改进的布谷鸟搜索算法及其应用研究[D]. 重庆：西南大学，2015.
XUE Y G. The improved cuckoo search algorithms and their application researches [D]. Chongqing: Southwest University, 2015.

[11] YANG X S, DE B S. Engineering optimisation by cuckoo search[J]. International Journal of Mathematical Modelling and Numerical Optimisation, 2010, 1(4): 330-343.

Gaussian Perturbating and Dynamic Grouping for Cuckoo Search

XUE Yige[1], DENG Huiwen[2]

(1. College of Information Engineering, Wenzhou Business College, Wenzhou Zhejiang 325035;
2. College of Computer & Information Science, Southwest University, Chongqing 400715, China)

Abstract: [**Purposes**]Cuckoo search (CS) is a new heuristic intelligent algorithm, which is easy to fall into local convergence, convergence speed is not high enough and the solution accuracy is not fast enough. Aiming at the shortcomings of the original cuckoo algorithm in solving the optimization problem, gaussian perturbating and Dynamic grouping for cuckoo search (GPDGCS) is proposed. [**Methods**]GPDGCS algorithm uses Gaussian perturbation and dynamic grouping strategy in the process of solving the original cuckoo algorithm, and avoids the local optimization of the algorithm to a certain extent. [**Findings**]The results show that the GPDGCS algorithm has higher convergence speed and accuracy than the original cuckoo algorithm through 6 typical test functions. [**Conclusions**]The GPDGCS algorithm can avoid the algorithm getting into local optimum to a certain extent.

Keywords: cuckoo search algorithm; gaussian perturbation; dynamic grouping

(责任编辑 许 甲)