DOI:10.3969/j.issn.1673-4785.201409042
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20150630.1559.004.html

具有动态惯性权重的布谷鸟搜索算法

周欢[1],李煜[2]
(1.河南大学 商学院,河南 开封 475004; 2.河南大学 管理科学与工程研究所,河南 开封 475004)

摘 要:为提高布谷鸟搜索算法的搜索能力和寻优精度,提出一种具有动态惯性权重的布谷鸟搜索算法。该算法引入动态惯性权重改进鸟窝位置的更新方式,依据动态惯性权重值保留上代鸟窝的最优位置并进行下一代位置更新,从而有效平衡种群探索能力和开发能力之间的关系。并利用特征方程对改进算法进行了收敛性分析。仿真实验结果表明,与基本布谷鸟搜索算法、粒子群算法和蚁群算法相比,改进后的布谷鸟搜索算法能显著减少迭代次数和运行时间,有效提高算法的收敛速度和收敛精度。

关键词:布谷鸟搜索算法;函数优化;莱维飞行;动态惯性权重;种群规模;收敛性;复杂度;参数选取
中图分类号: TP301.6 **文献标志码**:A **文章编号**:1673-4785(2015)04-0645-07

中文引用格式:周欢,李煜. 具有动态惯性权重的布谷鸟搜索算法[J]. 智能系统学报, 2015, 10(4): 645-651.
英文引用格式:ZHOU Huan, LI Yu. Cuckoo search algorithm with dynamic inertia weight[J]. CAAI Transactions on Intelligent Systems, 2015, 10(4): 645-651.

Cuckoo search algorithm with dynamic inertia weight

ZHOU Huan[1], LI Yu[2]
(1.School of Business Administration, He´nan University, Kaifeng 475004, China; 2.Research Institute of Management Science and Engineering, He´nan University, Kaifeng 475004, China)

Abstract: In order to improve the search ability and optimization accuracy of cuckoo search algorithm, the cuckoo search with dynamic inertia weight is proposed. By utilizing the dynamic inertia weight, the improved cuckoo search updates the next nest position based on the former best nest position that has been saved with dynamic inertia weight, which can well balance the relation between population exploration and development capabilities. This paper also has a convergence analysis of the improved cuckoo search by the characteristic equation. The performance of the new method is compared with the basic cuckoo search, particle swarm optimization, ant colony optimization and other algorithms, showing that the improved cuckoo search algorithm can significantly reduce the number of iterations and running time, and can effectively improve the convergence speed and convergence precision.

Keywords: cuckoo search algorithm; function optimization; Lévy flight; dynamic inertia weight; population size; convergence; complexity; parameter selection

布谷鸟搜索算法是由剑桥大学学者 Yang 和拉曼工程大学 Deb 于 2009 年基于对布谷鸟寻窝寄生幼卵行为的模拟,提出的一种新的全局优化算法,即 Cuckoo Search(CS)算法[1]。由于其具有控制参数少、寻优能力强和简单易实施等特点,现已成功应用于解决函数优化问题[2]、工程结构优化问题[3]、TSP 问题[4]和 PID 控制器调整[5]等多个领域。此外,通过改进算法参数提高算法性能也成为一个重要的研究领域。Walton 对布谷鸟搜索算法的步长进行改进,增加了个体间的信息交流[6]。Zheng 在鸟窝位置选择更新方式中加入了高斯扰动项,进而提出基于高斯扰动的布谷鸟搜索算法[7]。苏芙华等通过引入动态局部搜索技术,进一步提高了布谷鸟搜索算法的性能[8]。针对算法在求解复杂函数时,后期

收稿日期:2014-09-30. **网络出版日期**:2015-06-30.
基金项目:河南省科技攻关重点基金资助项目(122102210201);河南大学研究生教育综合改革基金资助项目(Y1427056).
通信作者:李煜.E-mail:lyhenu@163.com.

收敛精度不高、迭代次数多、运算时间长等问题,本文提出一种基于函数优化的具有动态惯性权重的布谷鸟(cuckoo search with dynamic inertia weight,简称 WCS)算法,通过加入动态惯性权重对鸟窝位置进行更新,有效平衡局部搜索和全局搜索之间的关系。实验结果表明,WCS 算法能够有效提高算法的收敛精度,减少算法的运行时间,具有较好的优化性能。

1 基本布谷鸟算法

布谷鸟算法源于对布谷鸟繁育行为的模拟,是新型有效的全局优化算法。其原理是将布谷鸟所选宿主的鸟窝映射为空间中的解,用宿主鸟巢所在位置的优劣表示解的适应度值,布谷鸟搜索和选择鸟窝的过程就是算法的搜索和优化过程[9]。

布谷鸟算法基于 3 个理想规则[1]:

规则 1 每只布谷鸟每次产一个卵,并随机选择鸟巢孵化它;

规则 2 在随机选择的一组鸟巢中,最好的鸟巢被保留至下一代;

规则 3 可选择的寄生巢的数量是固定的,寄生巢主人发现外来鸟蛋的概率为 P_a,其中 $0 \leqslant P_a \leqslant 1$。

基于这 3 个理想规则,Yang 等[1]采用式(1)对下一代鸟巢位置 $X^{(t+1)}$ 进行更新:

$$X_i^{(t+1)} = X_i^{(t)} + \alpha \oplus \text{Levy}(\lambda) \tag{1}$$

式中:t 为当前迭代次数,α 为步长控制量,在大多数情况下,取 $\alpha = 1$。$\oplus$ 为点对点乘法,Levy(λ)为随机搜索路径,属于随机行走,采用莱维飞行(Levy)机制[10],其行走的步长满足一个重尾的稳定分布。式(1)本质为随机行走方程,一般情况下,一个随机行走为一个马尔可夫链,其未来位置取决于当前位置(式(1)的第 1 项)和转移概率(第 2 项)[11]。

基本布谷鸟搜索算法,按照更新式(1)对下一代的鸟窝位置进行更新,并且计算目标函数的适应度值,如果优于上一代适应度值,则保留到下一代,否则保持原来位置不变。随机产生的数值 r 服从 0 到 1 均匀分布,并将其与鸟巢主人发现外来鸟蛋的概率 P_a 相比较,若 $r > P_a$,则重新产生一组初始解向量,否则保持不变。

2 具有动态惯性权重的布谷鸟搜索算法

2.1 动态惯性权重的选择

动态惯性改进策略是一种控制种群探索能力和开发能力的机制,能够有效提高算法的搜索能力,平衡局部搜索和全局搜索之间的关系。Shi 和 Eberhart[12]将惯性权重引入粒子群算法,作为一种取代速度钳制操作的控制机制,通过惯性权重的调整来控制算法的局部和全局寻优能力。

动态变化惯性权重主要有以下几类:

线性递减 w 的取值从较大值线性递减到一个较小的值[13]:

$$w(t) = (w(0) - w(n_t))\frac{n_t - t}{n_t} + w(n_t),$$
$$w(0) > w(n_t)$$

式中:n_t 为最大迭代次数,$w(0)$ 为初始惯性权重值,$w(n_t)$ 为最终惯性权重值,$w(t)$ 为第 t 次的惯性权重值,线性权重值一般从 0.9 递减为 0.4。

非线性递减 w 从较大值非线性递减到一个较小值[14]:

$$w(t+1) = \frac{(w(t) - 0.4)(n_t - t)}{n_t + 0.4}$$

模糊调整 由 Eberhart 和 Shi(1998)采用模糊系统提出的具有动态调整特征的自适应惯性权重[15],但其参数较多,增加了算法的复杂度和实现难度。

随机调整 Eberhart 等[16]提出一种动态惯性权重的方法,每次迭代都随机选取高斯分布中的一个惯性权重值,较大的惯性权重和较小的惯性权重随机出现。较大的惯性权重可以避免算法在初期陷入局部最优,同时较小的惯性权重可以解决后期收敛精度不高等问题[17]。

基于以上分析,本文选取动态惯性权重中的随机调整惯性权重对布谷鸟算法进行改进。随机调整的惯性权重服从高斯分布即:$w \sim N(\theta, \sigma)$。较大的惯性权重和较小的惯性权重都会随机出现,使得算法能够实现局部搜索和全局搜索的平衡。随机惯性策略 w 的更新公式为

$$w = r_{\min} + (r_{\max} - r_{\min}) \cdot \text{normrnd}() + \sigma \text{randn}()$$

式中:$r_{\min}$ 表示惯性权重的最小值,$r_{\max}$ 表示惯性权重的最大值,normrnd() 为服从均匀分布的随机数,randn() 为服从正态分布的随机数,σ 用来控制惯性权重与期望值之间的偏差程度。

因此,具有动态惯性权重的布谷鸟搜索算法采用动态惯性权重中的随机调整的惯性权重改进鸟窝位置更新方式,WCS 算法实现流程如下:

1)确定目标函数 $f(X)$,设置优化函数的定义域 $[-L, L]$,维度 d,发现概率 P_a 和种群数量 n。初始化种群,随机并产生 d 维向量,即 $\boldsymbol{X} = (x_1, x_2, \cdots, x_d)^{\mathrm{T}}$,$n$ 个鸟窝的初始位置为:X_i ($i = 1, 2, \cdots, n$);

2)计算每个鸟窝的目标函数值,并记录当前鸟窝位置的最优解;

3)保留上代最优鸟窝位置,产生随机动态惯性

权重 $w \sim N(\theta, \sigma)$,采用加入动态惯性权重策略的位置更新公式(2)对一代鸟窝位置进行更新;

$$X_i^{(t+1)} = w \cdot X_i^{(t)} + \alpha \oplus \text{Levy}(\lambda) \quad (2)$$

4)将现有鸟窝位置与上一代鸟窝位置进行比较,若现在位置较好,则将其作为当前的最优位置,否则最优位置不变;

5)用一个随机数 r,作为鸟窝主人发现外来鸟蛋的可能性,与发现概率 P_a 进行比较,若 $r \leqslant P_a$,则记录当前最优值;若 $r > P_a$,则返回 2),随机改变鸟窝位置,得到一组新的鸟窝位置;

6)输出全局最优值。

2.2　动态惯性权重的变化范围对优化效果的影响

本文参考文献[18]对动态惯性权重范围的划分进行取值,使用表 2 中函数 f_1 和函数 f_2 观察不同惯性权重的变化范围对优化效果的影响。参数设置:种群大小为 25,每次运行最大迭代次数为 50,对每个函数优化 50 次。实验结果如表 1 所示。表中,1 表示惯性权重范围 0~0.2,2 表示0.2~0.9,3 表示 0.9~1.2。实验结果表明,当取值在 0.9~1.2 时,算法找到最优结果的次数较少;当取值在 0.2~0.9 之间时,算法能够找到最优结果,但是找到最优解的概率不稳定;当取值在 0~0.2 时,算法能够找到最优结果,准确率较高。因此,随机调整的惯性权重服从高斯分布,且 θ 取值为 0~0.2, σ 取值足够小以确保 w 不会显著大于 1。

表 1　不同变化范围的惯性权重的优化效果

Table 1　Optimization effect of inertia weight with different changing ranges

f	惯性权重范围	1	2	3
f_1	平均迭代次数	2.4	3.4	14.9
	准确率	1	1	2%
f_2	平均迭代次数	2.6	8.5	2.9
	准确率	1	6%	0

2.3　算法的时间复杂度分析

算法的时间复杂性可以从侧面反映算法收敛速度的快慢,如果一个算法收敛,但需要无穷的迭代时间,那么这个算法不具有有效性。基本布谷鸟算法输入的种群规模为 N,求解目标函数为 $f(x)$,最大迭代次数为 n,根据基本布谷鸟算法的流程,假设产生均匀分布随机数的时间为 ξ_1,下一代鸟窝位置由上代鸟窝位置和 Levy 飞行机制得到,计算目标函数的时间复杂度为 $O(N(\xi_1 n + f(n))) = O(n + f(n))$,更新式(1)产生一组新解的时间为 ξ_2,新的位置与上代位置进行比较的执行时间为 ξ_3,若新位置较好,替换上代位置的执行时间为 ξ_4,在第 1 个新解产生阶段,时间复杂度为 $O(N(\xi_2 n + f(n) + \xi_3 + \xi_4 n)) = O(n + f(n))$。在第 2 个新解阶段,生成新解的执行时间为 ξ_5,时间复杂度为 $O(N(\xi_5 n + f(n) + \xi_3 + \xi_4 n) = O(n + f(n))$。记录最优鸟巢位置的时间复杂度为 $O(N(\xi_3 + \xi_4 n) = O(n)$,则对于每一代找到最优解的总的时间复杂度为 $T(n) = 3O(n + f(n)) + O(n) = O(n + f(n))$[19]。

同理,分析改进布谷鸟算法的流程,位置更新公式中加入动态惯性权重,设产生动态惯性权重 w 的执行时间为 ξ,种群数量、目标函数和最大迭代次数不变。在第 1 个新解阶段,时间复杂度为 $O(N(\xi_1 n + \xi_2 n + f(n) + \xi_3 + \xi_4 n)) = O(n + f(n))$,第 2 个新解阶段,时间复杂度为 $O(N(\xi_1 n + \xi_5 n + f(n) + \xi_3 + \xi_4 n)) = O(n + f(n))$。因此,更新下一代鸟窝位置的最终时间复杂度不变,仍为 $T(n) = O(n + f(n))$,与基本布谷鸟算法一致,改进后的算法没有增加时间复杂度。

3　WCS 算法的收敛性分析

标准布谷鸟算法是一种随机算法,文献[20]通过建立 Markov 链模型,利用随机算法收敛的条件证明了标准 CS 算法的收敛性,为 CS 算法全局收敛提供了理论依据。

文献[21]和[22]利用差分方程分别对蝙蝠算法和粒子群算法的收敛性进行了深入分析,本文采取同样的方法,通过建立差分方程对改进的布谷鸟算法进行收敛性分析。改进布谷鸟算法采用式(2)进行下一代鸟窝位置更新,式(2)表示第 i 个鸟巢在 $t+1$ 时刻的位置, α 为步长控制量,Levy(λ)是随机搜索步长,即

$$\text{Levy}(\lambda) \sim 0.01 \times \frac{u}{|v|^{1/\beta}} \times (g_{\text{best}} - X_i^{(t)}) \quad (3)$$

式中:u 和 v 服从均匀分布,$u \sim N(0, \sigma_u^2)$,$v \sim N(0, \sigma_v^2)$, α 通常取值为 1,为简化计算,设 $0.01 \times \frac{u}{|v|^{1/\beta}}$ 为一常量 r_1, g_{best} 为第 t 代个体鸟巢最优位置,记为 g_b,第 $t+1$ 代鸟巢最优位置记为 p_b,式(2)可化为

$$X^{(t+1)} = w \cdot X^{(t)} + r_1 \cdot (g_b - X^{(t)}) \quad (4)$$

$$X^{(t+2)} = w \cdot X^{(t+1)} + r_1 \cdot (p_b - X^{(t+1)}) \quad (5)$$

由式(4)和式(5)可得:

$$X^{(t+2)} + (1 + r_1 - w) \cdot X^{(t+1)} + (r_1 - w) \cdot X^{(t)} = r_1 p_b + r_1 g_b \quad (6)$$

式(6)是一个二阶的常系数非齐次差分方程,求解其差分方程的特征方程得:

$$\lambda^2 + (1 + w - r_1)\lambda + w - r_1 = 0$$

求解特征方程,得到 $\triangle = (1 + w - r_1)^2 - 4(w - r_1)$,

因为 $\triangle=(w-r_1-1)^2\geqslant 0$,所以只需要考虑以下 2 种情况:

(a)当 $\triangle=(w-r_1-1)^2=0$ 时,即 $w-r_1-1=0$。λ_1 和 λ_2 为特征方程的 2 个根,若 λ_1 和 λ_2 为实数,$\|\lambda_1\|$、$\|\lambda_2\|$ 分别表示 λ_1和 λ_2的绝对值,若为复数,则表示 λ_1和 λ_2 的模值。

由 $\triangle=0$ 得:$\lambda_1=\lambda_2=-1$。此时 $X^{(t)}=(A_0+A_1 t)\lambda^t$,$A_0$、$A_1$ 为待定系数,计算可得:

$$\begin{cases}A_0=x(0)\\A_1=(r_1-w-1)x(0)-r_1 g_b\end{cases}$$

(b)当$\triangle>0$ 时,此时 $X^{(t)}=A_0+A_1\lambda_1^t+A_2\lambda_2^t$,$A_0$、$A_1$ 和 A_2 为待定系数,计算得到:

$$\begin{cases}\lambda_1=\dfrac{-(1+w-r_1)+\sqrt{\triangle}}{2}\\\lambda_2=\dfrac{-(1+w-r_1)-\sqrt{\triangle}}{2}\end{cases}$$

当 $t\to\infty$ 时,有极限且趋向于有限值,表示迭代收敛,由此可知,若上面 2 种情况中 $X^{(t)}$ 收敛,则需满足条件 $\|\lambda_1\|<1$,$\|\lambda_2\|<1$[19]。当 $\max(\|\lambda_1\|,\|\lambda_2\|)=1$,则有:

$$\lim_{t\to\infty}X^{(t)}=A_0+A_1+A_3,$$

$$\lim_{t\to\infty}X^{(t)}=A_0+A_3 \text{ 或者} \lim_{t\to\infty}X^{(t)}=A_0+A_2,$$

粒子的轨迹收敛[20]。

因此,当$\triangle=0$ 时,收敛区域为 $w-r_1-1=0$;当$\triangle>0$ 时,收敛区域需要满足:$-1<r_1-w<1$。综合上面 2 种情况,收敛区域为$-1\leqslant r_1-w<1$。

4 仿真实验

为评估 WCS 算法的性能,采用 5 个典型的测试函数进行验证。运行环境:CPU 为 B940@ 2.00 GHz,内存 2 GB,Matlab R2013a 编程环境。其中,函数f_1~f_5的最优值分别为 1、3 600、0、0 和−3,测试函数 f_1~f_5的数学表达式和自变量的取值范围如下:

$$f_1=0.5-\frac{\sin\sqrt{x^2+y^2}-0.5}{1-0.001(x^2+y^2)}\quad x,y\in[-4,4]$$

$$f_2=\left[\frac{3}{0.05+(x^2+y^2)}\right]^2+(x^2+y^2)^2$$

$$x,y\in[-5.12,5.12]$$

$$f_3=10\cos(2\pi x)-10\cos(2\pi y)-x^2-y^2-20$$

$$x,y\in[-5.12,5.12]$$

$$f_4=\frac{1}{4\,000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{i}\right)+1$$

$$x\in[-600,600]$$

$$f_5=-[1+(x+y+1)^2(19-14x+3x^2-14y+6xy+3y^2)][30+(2x-3y)^2(18-32x+12x^2+48-36xy+27y^2)]$$

$$x,y\in[-2,2]$$

4.1 WCS 算法与 CS 算法的比较

算法参数设置为:种群规模 $n=25$,发现概率 $P_a=0.25$。惯性权重从高斯分布中取值并随机调整。每个函数分别进行 30 次独立仿真实验,平均时间为算法找到最优解所用时间的平均值,其单位为秒。平均迭代次数为算法收敛到最优解的代数的平均值。进化代数为算法找到最优解的最大迭代次数的最小值。准确率为算法找到最优解的次数与运行总次数的百分比。WCS 算法与 CS 算法比较,如图 1~6 所示,仿真运算结果如表 2 所示。

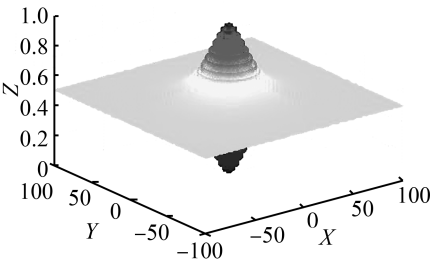

图 1 f_1图像

Fig.1 Image of f_1

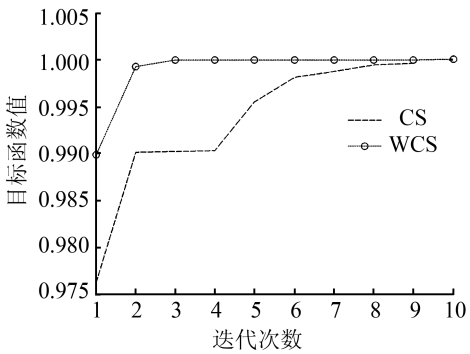

图 2 f_1 算法收敛状态对比图

Fig.2 Convergence state of CS and WCS on f_1

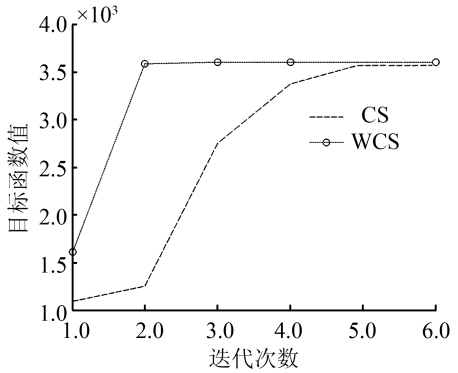

图 3 f_2 算法收敛对比图

Fig.3 Convergence state of CS and WCS on f_2

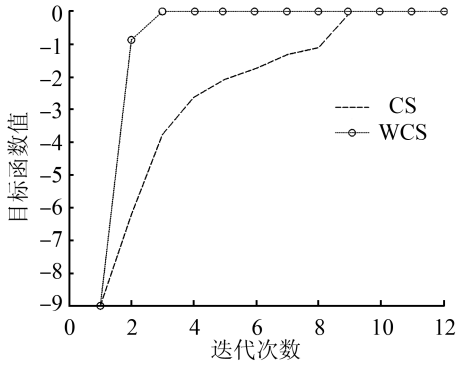

图 4　f_3 算法收敛对比图

Fig.4　Convergence state of CS and WCS on f_3

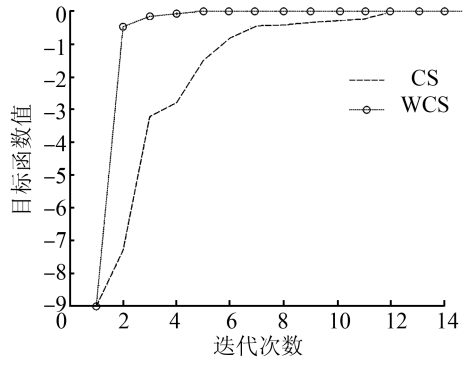

图 5　f_4 算法收敛对比图

Fig.5　Convergence state of CS and WCS on f_4

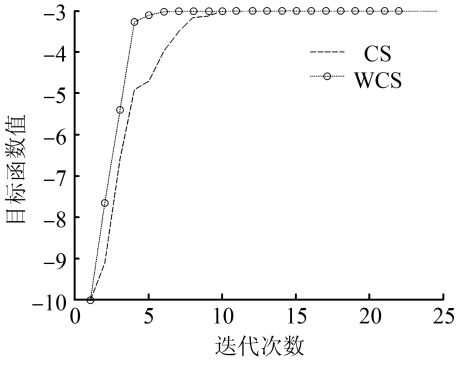

图 6　f_5 算法收敛对比图

Fig.6　Convergence state of CS and WCS on f_5

1) Schaffer 函数

$$f_1(x,y) = 0.5 - \frac{\sin\sqrt{x^2 + y^2} - 0.5}{1 + 0.001(x^2 + y^2)}$$

其中, x, $y \in [-4,4]$,函数最优解为 1,最优解位置为(0,0),运行 WCS 算法很快找到最优解 1。函数图像如图 1 所示,算法收敛状态对比见图 2,函数运算结果见表 2。

2) Needle-in-haystack 函数

$$f_1(x,y) = \left[\frac{3}{0.05 + (x^2 + y^2)}\right]^2 + (x^2 + y^2)^2$$

其中, x, $y \in [-5.12,5.12]$,该函数是多峰函数,有 4 个局部极值点分别为:(-5.12, 5.12)、(-5.12, -5.12) 、(5.12,5.12)和(5.12,-5.12),函数的全局最优值 3 600,最优解位置为(0,0)。运行 WCS 算法很快就可以收敛到最优解,函数 f_2 收敛性状态对比如图 3 所示,其余函数 f_3 ~ f_5 的收敛对比见图 4~6,运算结果见表 2。

以上函数收敛图表明,改进后的算法和基本算法相比,迭代次数更少,收敛速度更快,收敛精度更高,具有更好的优化性能。

表 2　WCS 算法与 CS 算法结果比较测试函数

Table 2　Comparison of WCS with CS

函数		平均时间/s	平均迭代次数	进化代数	准确率/%
f_1	CS	9.996 4	11.233 3	2 500	83.33
	WCS	0.170 2	4.733 33	50	100.00
f_2	CS	1.021 6	4.966 67	300	13.33
	WCS	0.093 5	3.666 67	20	90.00
f_3	CS	2.218 9	12.766 7	600	30.00
	WCS	0.315 4	7.9	70	83.33
f_4	CS	3.951 5	19.966 7	1 100	33.33
	WCS	0.848 6	9.433 33	200	93.33
f_5	CS	0.045 2	4.466 7	50	100
	WCS	0.043 3	5.966 67	40	100

表 2 运算结果表明,对于 5 个测试函数,改进后的算法找到最优解的次数更多,时间更短,相比于基本算法找到最优解的准确率更高。当基本 CS 算法面对较为复杂的函数优化问题时,如 f_1,增加迭代次数为 2 500,基本算法运行时间是改进后算法的 58.7 倍,收敛到最优解的迭代次数是改进后算法的 2.4 倍,准确率只有改进后算法的 83%。因此,改进后的算法可以更好地解决函数优化问题。

4.2　WCS 算法与其他算法的比较

用以上函数对 WCS 算法与 CS 算法、粒子群算法(PSO)、蚁群算法(ACO)、引力搜索算法(GSA)和蝙蝠算法(BA)进行性能测试,运算结果如表 3 所示。

其中,为了更加公平地测试算法性能,每个算法独立运行 50 次,种群数量都为 50,迭代次数 1 000。CS 算法和 WCS 算法参数设置为:发现概率 $P_a = 0.25$。PSO 参数设置为 $w = 0.729$, $c_1 = c_2 = 1.494\ 45$[23]。蚁群算法参数设置为: $\alpha = \beta = 1$, $r = 0.5$, $\rho = 0.7$, $Q = 1$[24]。引力搜索算法参数设置为: $T = 1\ 000$, $G_0 = 100$, $\alpha = 50$。蝙蝠算法参数设置为: $A_i = 0.25$, $r_i = 0.5$, $\alpha = \gamma = 0.9$[25]。

实验结果表明,对于函数 f_1 ~ f_5,相比于其他算法,WCS 算法能够很快找到最优解。蚁群算法表现次之,对于 f_1 ~ f_4 可以收敛到最优解,但对 f_5 求解精度

不够。CS 算法求解的稳定性不高。对于函数 f_1 和 f_2,WCS 算法和蚁群算法都有很好的收敛性,能快速找到最优解;对于函数 f_3,WCS 算法和蚁群算法性能明显优于其他算法;对于函数 f_4,WCS 算法和蚁群算法能够收敛到最优解;对于函数 f_5,CS 算法、WCS 算法和新型蝙蝠算法优化性能明显优于其他算法。

表 3 WCS 算法与其他算法性能比较

Table 3 Performance comparison of WCS with other algorithms

f	算法	最优值	平均值	最差值	标准差
f_1	CS	0.999 974	0.997 223	0.990 284	0.003 559
	WCS	1	1	1	0
	PSO	1	−0.999 87	−0.996 87	0.000 619
	ACO	1	1	1	0
	GSA	1	0.993 167	0.990 284	0.003 921
	BA	1	0.997 667	0.990 280	0.004 193
f_2	CS	3.60×10^{3}	3.55×10^{3}	2.75×10^{3}	194.710 1
	WCS	3 600	3 600	3 600	0
	PSO	3.60×10^{3}	3.60×10^{3}	3.59×10^{3}	0.000 162
	ACO	3 600	3 600	3 600	0
	GSA	3.58×10^{3}	3.14×10^{3}	2.71×10^{3}	325.587 9
	BA	3 600	3 600	3 600	0
f_3	CS	−0.000 890	−0.202 307	−1.041 848	0.274 363
	WCS	0	0	0	0
	PSO	0	-1.69×10^{-5}	-5.08×10^{-4}	0.000 077
	ACO	0	0	0	0
	GSA	0	−0.047 840	−0.853 809	0.127 847
	BA	0	-2.82×10^{-7}	-4.64×10^{-6}	6.62×10^{-7}
f_4	CS	-6.67×10^{-9}	-2.52×10^{-4}	-6.28×10^{-3}	0.000 989
	WCS	0	0	0	0
	PSO	-2.36×10^{-13}	5.43×10^{-11}	-5.18×10^{-10}	7.99×10^{-11}
	ACO	0	0	0	0
	GSA	0	−0.006 084	−0.029 799	0.006 14
	BA	-3.58×10^{-12}	-3.99×10^{-10}	-1.41×10^{-9}	3.98×10^{-10}
f_5	CS	−3	−3	−3	0
	WCS	−3	−3	−3	0
	PSO	−3.000 000 1	−3.000 004	−3.000 019	4.98×10^{-8}
	ACO	−3	−3.167 406	−3.665 694	0.176 342
	GSA	−3	−3.010 484	−3.391 7	0.058 115
	BA	−3	−3	−3	0

4.3 种群大小对惯性权重选择的影响

算法参数的选择可能对算法性能产生重要影响,本文重点讨论种群大小对算法性能的影响。种群规模由 25 增加至 100,最大迭代次数为 300,其他参数设置不变,每个函数独立运行 50 次,1、2 和 3 分别表示惯性权重的取值范围 0~0.2、0.2~0.9 和0.9~1.2,4 表示基本布谷鸟算法。在 n 为 25 和 100 两种情况下,函数 f_1~f_3 找到最优解的达优率如表 4 所示。

由表 4 可得,种群规模增大后,CS 算法优化性能基本不变,函数 2 在种群规模增大后的达优率甚至出现减少的情况;惯性权重范围在 0.2~0.9 和 0.9~1.2 两个区间的算法优化性能并没有得到明显改善,最佳惯性权重范围仍然是 0~0.2,说明种群规模的改变,对算法收敛效果的影响不显著。

表 4 不同种群下算法优化效果

Table 4 Optimization effect of CS on different populations

f	n	1	2	3	4
f_1	25	1	0.9	0	0
	100	1	1	0	0
f_2	25	1	1	0.04	0.12
	100	1	1	0.08	0.08
f_3	25	1	1	0	0
	100	1	1	0	0

5 结束语

针对布谷鸟算法在求解函数最优化问题中存在的收敛度不高、收敛速度慢等问题,本文提出了一种改进方法:通过引入动态惯性权重改进鸟窝位置的更新方式,对动态惯性权重的取值范围进行分析,找到求解最优的惯性权重取值范围。并利用差分方程对改进算法进行了收敛性分析。研究发现改进后的算法通过动态惯性权重控制种群探索能力和开发能力,能够有效平衡局部搜索能力和全局搜索能力之间的关系。仿真实验结果表明,与基本布谷鸟算法相比,WCS 算法可以显著减少迭代次数,缩短运行时间;与其他算法相比,WCS 算法也有较强的收敛性和鲁棒性。最后,对算法参数进行了讨论,当种群规模增大后,仍需惯性权重的调节,且惯性权重的选择范围不变,进一步说明,加入动态惯性权重后的算法具有更好的优化性能。

参考文献:

[1] YANG Xinshe, DEB S. Cuckoo search via Lévy flights [C]//World Congress on Nature & Biologically Inspired Computing. Coimbatore, India, 2009: 210-214.

[2]李煜, 马良. 新型元启发式布谷鸟搜索算法[J]. 系统工程, 2012, 30(8): 64-69.
LI Yu, MA Liang. A new metaheuristic cuckoo search algorithm[J]. Systems Engineering, 2012, 30(8): 64-69.

[3]陈乐, 龙文. 求解工程结构优化问题的改进布谷鸟搜索算法[J]. 计算机应用研究, 2014, 31(3): 679-683.
CHEN Le, LONG Wen. Modified cuckoo search algorithm for solving engineering structural optimization problem[J]. Application Research of Computers, 2014, 31(3): 679-683.

[4] OUAARAB A, AHIOD B, YANG Xinshe. Discrete cuckoo search algorithm for the travelling salesman problem[J]. Neural Computing and Applications, 2014, 24(7/8): 1659-1669.

[5] SETHI R, PANDA S, SAHOO B P. Cuckoo search algorithm based optimal tuning of PID structured TCSC control-

ler[M]//JAIN L C, BEHERA H S, MANDAL J K, et al. Computational Intelligence in Data Mining-Volume 1. Odisha: Springer, 2015: 251-263.

[6] WALTON S, HASSAN O, MORGAN K, et al. Modified cuckoo search: a new gradient free optimisation algorithm [J]. Chaos, Solitons and Fractals, 2011, 44(9): 710-718.

[7] ZHENG Hongqing, ZHOU Yongquan. A novel cuckoo search optimization algorithm based on Gauss distribution [J]. Journal of Computational Information Systems, 2012, 8(10): 4193-4200.

[8]苏芙华, 刘云连, 伍铁斌. 求解无约束优化问题的改进布谷鸟搜索算法[J]. 计算机工程, 2014, 40(5): 224-227, 233.
SU Fuhua, LIU Yunlian, WU Tiebin. Modified cuckoo search algorithm for solving unconstrained optimization problem[J]. Computer Engineering, 2014, 40(5): 224-227, 233.

[9]龙文, 陈乐. 求解约束化工优化问题的混合布谷鸟搜索算法[J]. 计算机应用, 2014, 34(2): 523-527.
LONG Wen, CHEN Le. Hybrid cuckoo search algorithm for solving constrained chemical engineering optimization problems[J]. Journal of Computer Applications, 2014, 34(2): 523-527.

[10] VISWANATHAN G M, AFANASYEV V, BULDYREV S V, et al. Lévy flights in random searches[J]. Physica A: Statistical Mechanics and its Applications, 2000, 282(1/2): 1-12.

[11]邓鑫洋, 邓勇, 章雅娟, 等. 一种信度马尔科夫模型及应用[J]. 自动化学报, 2012, 38(4): 666-672.
DENG Xinyang, DENG Yong, ZHANG Yajuan, et al. A belief Markov model and its application[J]. Acta Automatica Sinica, 2012, 38(4): 666-672.

[12] SHI Yuhui, EBERHART R. A modified particle swarm optimizer[C]//IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on Evolutionary Computation Proceedings. Anchorage, USA, 1998: 69-73.

[13] SHI Yuhui, EBERHART R C. Empirical study of particle swarm optimization[C]//Proceedings of the 1999 Congress on Evolutionary Computation. Washington DC, USA, 1999, 3: 1945-1949.

[14] PERAM T, VEERAMACHANENI K, MOHAN C K. Fitness-distance-ratio based particle swarmoptimization[C]//Proceedings of the 2003 IEEE Swarm Intelligence Symposium. Indianapolis, USA, 2003: 174-181.

[15] SHI Yuhui, EBERHART R C. Fuzzy adaptive particle swarm optimization[C]//Proceedings of the 2001 Congress on Evolutionary Computation. Seoul, Korea, 2001: 101-106.

[16] EBERHART R C, SHI Yuhui. Tracking and optimizing dynamic systems with particle swarms[C]//Proceedings of the 2001 Congress on Evolutionary Computation. Seoul, Korea, 2001: 94-100.

[17] ZHANG Liping, YU Huanjun, HU Shangxu. A new approach to improve particle swarm Optimization[C]//Genetic and Evolutionary Computation—GECCO 2003. Berlin Heidelberg, Germany, 2003: 134-139.

[18]王俊伟, 汪定伟. 粒子群算法中惯性权重的实验与分析[J]. 系统工程学报, 2005, 20(2): 194-198.
WANG Junwei, WANG Dingwei. Experiments and analysis on inertia weight in particle swarm optimization[J]. Journal of Systems Engineering, 2005, 20(2): 194-198.

[19]张永韡, 汪镭, 吴启迪. 动态适应布谷鸟搜索算法[J]. 控制与决策, 2014, 29(4): 617-622.
ZHANG Yongwei, WANG Lei, WU Qidi. Dynamic adaptation cuckoo search algorithm[J]. Control and Decision, 2014, 29(4): 617-622.

[20]王凡, 贺兴时, 王燕, 等. 基于 CS 算法的 Markov 模型及收敛性分析[J]. 计算机工程, 2012, 38(11): 180-182, 185.
WANG Fan, HE Xingshi, WANG Yan, et al. Markov model and convergence analysis based on cuckoo search algorithm[J]. Computer Engineering, 2012, 38(11): 180-182, 185.

[21]李枝勇, 马良, 张惠珍. 蝙蝠算法收敛性分析[J]. 数学的实践与认识, 2013, 43(12): 182-190.
LI Zhiyong, MA Liang, ZHANG Huizhen. Convergence analysis of bat algorithm[J]. Mathematics in Practice and Theory, 2013, 43(12): 182-190.

[22]刘洪波, 王秀坤, 谭国真. 粒子群优化算法的收敛性分析及其混沌改进算法[J]. 控制与决策, 2006, 21(6): 636-640.
LIU Hongbo, WANG Xiukun, TAN Guozhen. Convergence analysis of particle swarm optimization and its improved algorithm based on chaos[J]. Control and Decision, 2006, 21(6): 636-640.

[23] EBERHART R C, SHI Y. Comparing inertia weights and constriction factors in particle swarm optimization[C]//Proceedings of the 2000 Congress on Evolutionary Computation. La Jolla, Germany, 2000: 84-88.

[24]马良. 基于蚂蚁算法的函数优化[J]. 控制与决 策, 2002, 17(增刊): 719-722.
MA Liang. Ant algorithm based function optimization[J]. Control and Decision, 2002, 17(S): 719-722.

[25]李枝勇, 马良, 张惠珍. 多目标 0-1 规划问题的蝙蝠算法[J]. 智能系统学报, 2014, 9(6): 672 -676.
LI Zhiyong, MA Liang, ZHANG Huizhen. Bat algorithm for the multi-objective 0-1 programming problem[J]. CAAI Transactions on Intelligent Systems, 2014, 9(6): 672-676.

作者简介:

周欢,1990 年生,女,硕士研究生,主要研究方向为智能优化、电子商务。

李煜,1969 年生,女,教授,博士,主要研究方向为智能优化、电子商务、物流管理。

[责任编辑:孟玮]