一种新的自适应布谷鸟搜索算法

钱伟懿[1] 候慧超[1] 姜守勇[2]
(渤海大学数理学院 锦州 121000)[1] (东北大学信息科学与工程学院 沈阳 110819)[2]

摘 要 为提高布谷鸟搜索(cuckoo search)算法(CS)的局部与全局搜索能力和收敛速度,提出了一种新的自适应布谷鸟算法。在该算法中,提出一种自适应参数控制策略来动态地调整CS中的步长因子,以增强CS的搜索性能。另外,把类似差分进化算法变异策略引入到CS中,以进一步提高CS的种群多样性。仿真实验表明,改进的CS算法的优化性能得到了明显改善。

关键词 布谷鸟搜索算法,莱维飞行,自适应,变异
中图法分类号 TP18 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.07.058

New Self-adaptive Cuckoo Search Algorithm

QIAN Wei-yi[1] HOU Hui-chao[1] JIANG Shou-yong[2]
(School of Mathematics and Physics, Bohai University, Jinzhou 121000, China)[1]
(College of Information Science and Engineering, Northeastern University, Shenyang 110819, China)[2]

Abstract In order to improve the local and global search ability of cuckoo search algorithm(CS) and its convergence rate, a new self-adaptive cuckoo search algorithm was proposed. In this algorithm, a self-adaptive parameter control strategy is used to adjust the step size of CS, thereby enhancing the search ability of CS. In addition, a mutation technique which is similar to differential evolution algorithm is utilized to guarantee the CS diversity. Experimental results show that the proposed algorithm is much more effective.

Keywords Cuckoo search algorithm, Levy flight, Self-adaptation, Mutation

1 引言

优化问题与人类的生存息息相关。近年来优化问题在日常生活及工程应用等领域层出不穷,并且越来越复杂。传统的数值计算方法在日益复杂的优化问题上暴露出自身的缺陷,在优化求解上越发显得力不从心。于是,研究人员转向依赖于启发式算法求解过于复杂的优化问题,诸如遗传算法[1]、粒子群算法[2]、差分进化算法[3]等。在许多复杂的优化问题上,这类启发式算法都能提供比传统的数值方法更好的解。

2009年,Yang和Deb提出了一种新的启发式算法——布谷鸟搜索(Cuckoo Search)算法(CS)[4]。该算法模拟了布谷鸟寻窝产卵的行为,并引入了一些鸟及果蝇的莱维飞行机制,使其能够快速有效地寻找到最优解。文献[4]表明,在多峰值优化问题上CS比PSO和GA的鲁棒性和遗传性更强,而且寻找到最优解的成功率更高。此外,CS算法已成功应用于多种工程优化问题,如结构优化问题[5]、工程优化问题[6]、铣削操作优化问题[7]、背包问题[9]及交通流量预测问题[9]等。

尽管CS算法有着结构简单、控制参数少等优点,但CS自身也存在搜索活性不强、收敛速度慢的缺陷。CS算法的步长因子是一个重要参数,它控制着CS算法的局部和全局探索能力。Walton等人建议使用随代数递减的步长因子来加速算法的收敛速度[10]。Valian等人研究了步长因子对算法性能的影响,采用动态调整的策略增加算法的性能[11]。郑洪清、周永权给出了基于最佳鸟巢位置的自适应调整步长的策略[12]。王利英等人给出了基于莱维飞行,利用粒子群优化算法原理及环状拓扑结构调整步长的策略[13]。此外,为提高算法的求解精度和收敛速度,Zhang等人提出将高斯分布引入CS算法中并取得了较好的效果[14]。杜利敏等人把共轭梯度算法与CS算法结合,提高了CS算法的收敛能力[15];目前还有许多学者对CS算法进行广泛研究[16-21]。由于该算法完全依赖于随机行走策略,因此很难保证它有较快的收敛速度,而且搜索精度也得不到保障。基于这两方面的不足,本文对CS算法做了两方面改进,1)给出了一个类似于差分进化算法的变异、交叉和选择策略,并与CS算法结合;2)把鸟巢位置与步长因子合为一个新的变量,在改进的CS算法中对新变量进行操作,这样不仅改善了鸟巢位置,同时也能动态调整步长因子,从而提高算法的全局和局部搜索能力。最后选择5个典型测试问题验证算法的性能,数值结果表明,所给出的算法具有较好的收敛速度和较强的搜索能力。

2 CS算法

Yang和Deb开发的CS算法是一种群智能优化方法,受

到稿日期:2013-09-25 返修日期:2013-12-21 本文受国家自然科学基金项目(11371071),辽宁省自然基金项目(20102003),辽宁省教育厅科学研究项目(L2013426)资助。

钱伟懿(1963—),男,博士,教授,主要研究方向为最优化理论与应用、智能计算,E-mail:qianweiyi2008@163.com;候慧超(1988—),女,硕士生,主要研究方向为智能计算。

布谷鸟的寄生孵育雏鸟的生物现象启发而来。同时，为了模拟布谷鸟的繁殖方式，应假定3条近似规则：

(1)每只布谷鸟一次只能下一个卵，并且随机选取一个鸟巢孵化它。

(2)拥有最好卵的鸟巢将被保留到下一代中。

(3)鸟巢的数目是固定的，宿主发现布谷鸟的卵的概率是 P_a。如果宿主发现布谷鸟的卵，它要么抛弃这个卵，要么抛弃这个鸟巢，并在新的位置新建一个鸟巢。

第(3)条表达了布谷鸟算法的一种选择机制：在 N 个鸟巢中，有概率为 P_a 的鸟巢会因新产生的卵而被更新。在CS算法中，$x_i^t=(x_{i1}^t,x_{i2}^t,\cdots,x_{in}^t)^{\mathrm{T}}$（$n$ 为优化问题的维数）表示第 i 个鸟巢在第 t 代的鸟巢位置。新的鸟巢位置(解) x_i^{t+1} 由莱维飞行产生。因此，布谷鸟的鸟巢位置更新公式由式(1)给出：

$$x_i^{(t+1)}=x_i^{(t)}+\alpha\oplus L(\lambda) \tag{1}$$

其中，α 是步长因子，根据优化问题的实际情况而定。符号$\oplus$表示点对点乘法。$L(\lambda)$表示服从参数 $\lambda(1<\lambda<3)$ 的莱维分布产生的一个随机搜索向量，即：

$$L(\lambda):u=t^{-\lambda},1<\lambda<3 \tag{2}$$

该概率分布的均值和方差都是无界的。该分布说明布谷鸟连续的位置变动恰恰形成了一个服从幂律分布的随机行走过程。Yang提出的位置更新策略可概括如下[14]：

$$x_i^{(t+1)}=x_i^{(t)}+\alpha\oplus L(\lambda) \tag{3}$$

其中，$L(\lambda)=0.01\cdot\frac{u}{|v|}(x_i^{(t)}-x_b^{(t)})$，$u$ 和 v 都服从正态分布，即 $u\sim N(0,\sigma_u^2)$，$v\sim N(0,1)$，其中

$$\sigma_u=\left[\frac{\Gamma(\lambda)\sin(0.5\pi(\lambda-1))}{2^{(\lambda-2)/2}\Gamma(0.5\lambda)(\lambda-1)}\right]^{1/(\lambda-1)}$$

$x_b^{(t)}$ 表示当前时刻 t 所存储的最优鸟巢位置，Γ 是标准的Gamma函数。

CS算法的具体过程表述如下：布谷鸟先在解空间里确定一定数量的鸟巢并产卵，根据卵的优劣存储当前最好的鸟巢。利用式(1)更新鸟巢的位置并产卵。宿主若发现自己鸟巢中有“外来者”则放弃该巢并另建新巢，否则就接受位置更新后鸟巢。对全部鸟巢进行评估，重新存储最好的鸟巢。多次重复以上过程，布谷鸟就能找到最好的鸟巢，从而保证后代繁衍。

3 自适应CS算法

3.1 自适应CS算法

步长因子 α 对CS算法至关重要。Valian[11]等人的研究表明，动态地调整步长因子，会提高算法的搜索能力，并能加快收敛速度。本文提出将 α 融入到优化问题中，即对鸟巢的位置增加一个新的变量 α，第 i 鸟巢位置表示如下：

$$\hat{x}_i=(x_i^{\mathrm{T}},\alpha_i)^{\mathrm{T}}=(x_{i1},x_{i2},\cdots,x_{in},\alpha_i)^{\mathrm{T}} \tag{4}$$

其中，x_i 表示真实的鸟巢位置。因此，鸟巢的位置更新式(3)可改写为：

$$\hat{y}_i^{(t)}=\hat{x}_i^{(t)}+\alpha_i^{(t)}\oplus L(\lambda) \tag{5}$$

其中，$L(\lambda)=0.01\cdot\frac{u}{|v|}(\hat{x}_i^{(t)}-\hat{x}_b^{(t)})$，$\hat{x}_b^{(t)}=(x_b^{(t)\mathrm{T}},\alpha_i^{(t)})^{\mathrm{T}}$。需要注意的是，增加的变量只用来自适应调整步长因子，而不参与到适应值的计算。

3.2 变异策略

DE/rand/2/bin差分策略如下：

$$V_i=x_{r1}+F(x_{r2}-x_{r3})+F(x_{r4}-x_{r5})$$

其中，$x_n(i=1,2,3,4,5)$为随机选取的5个位置，F 为压缩因子。为了进一步增强CS算法的探索能力和种群的多样性，类似于DE/rand/2/bin差分策略，本文给出一种新的变异策略来更新鸟巢的位置。在第 t 代中，随机选取4个与鸟巢 i 互不相同的鸟巢，即 $\hat{y}_{r1}^{(t)},\hat{y}_{r2}^{(t)},\hat{y}_{r3}^{(t)},\hat{y}_{r4}^{(t)}$，用于新建一个变异鸟巢并更新鸟巢 i 的位置。令

$$V_i=\hat{y}_{r1}^{(t)}+r1\cdot(\hat{y}_{r2}^{(t)}-\hat{y}_{r3}^{(t)})+r2\cdot(\hat{y}_b^{(t)}-\hat{y}_{r4}^{(t)}) \tag{6}$$

其中，$\hat{y}_b^{(t)}=(y_b^{(t)\mathrm{T}},\alpha_i^{(t)})^{\mathrm{T}}$，$r1$ 和 $r2$ 是$[0,1]$上的随机数，$y_b^{(t)}$ 表示当前的最优鸟巢位置，$V_i=(v_{i1},,v_{i2},\cdots,v_{in},v_{i(n+1)})^{\mathrm{T}}$，$v_{i(n+1)}$为步长因子。然后以一定概率决定变异鸟巢是否作为备选鸟巢，具体形式如下：

$$u_{ij}=\begin{cases}v_{ij}, & r3<CR\\ x_{ij}^{(t)}, & \text{else}\end{cases},j=1,2,\cdots,n \tag{7}$$

其中，CR 是一小于1的非负常数，$r3$ 是$[0,1]$上的随机数。令 $U_i=(u_{i1},u_{i2},\cdots,u_{in})^{\mathrm{T}}$。最后，评估备选鸟巢的好坏，确定是否用它来替换原来鸟巢的位置：

$$x_i^{(t+1)}=\begin{cases}U_i, & finess(U_i)<finess(x_i^{(t)})\\ x_i^{(t)}, & \text{else}\end{cases} \tag{8}$$

步长因子更新如下：

$$\alpha_i^{(t+1)}=\begin{cases}v_{i(n+1)}, & r4<P_i\\ \alpha_i^i, & \text{else}\end{cases} \tag{9}$$

其中，$r4$ 是$[0,1]$上的随机数，P_i 为步长因子更新概率。按如下规则选取 P_i：

若由式(6)生成的鸟巢位置好，取

$$P_i^{(t+1)}=P_i^{(t)}+\beta_0P_i^{(t)}(1-P_i^{(t)}) \tag{10}$$

否则，取

$$P_i^{(t+1)}=P_i^{(t)}-\beta_1P_i^{(t)}(1-P_i^{(t)}) \tag{11}$$

其中，$\beta_0,\beta_1\in(0,1)$。式(10)和式(11)表明若在当前步长因子下能够得到较好的鸟巢位置，则保留当前步长概率增大，否则更新步长因子的概率增大。

3.3 自适应CS算法的实现过程

本文所给出的自适应CS算法的基本流程如下：

(a)初始化CS算法相关参数：确定宿主鸟巢数目 N，发现概率 P_a，最大迭代次数 $Max\text{-}iter$，步长更新概率 P_0。初始化布谷鸟随机选取鸟巢的位置 $\hat{x}_i^{(0)}=(x_{i1}^{(0)},x_{i2}^{(0)},\cdots,x_{in}^{(0)},\alpha_i^{(0)})^{\mathrm{T}}$，计算各鸟巢的适应值，得到 $\hat{x}_b^{(0)}$，令 $t=0$。

(b)若满足终止条件，则停；否则，转向(c)；

(c)使用式(5)更新鸟巢的位置 $\hat{y}_i^{(t)}$，并获得 $y_b^{(t)}$；

(d)根据式(6)—式(9)进一步更新鸟巢的位置 $x_i^{(t+1)}$ 和步长因子 $\alpha_i^{(t+1)}$；

(e)若 $y_b^{(t)}$ 得到改进，则按式(10)更新 $P_i^{(t+1)}$，否则按式(11)更新 $P_i^{(t+1)}$；

(f)产生一个随机数 $r\in[0,1]$，如果 $r>P_a$，宿主将放弃自己的巢，并另建新巢，否则接受更新位置后的鸟巢；并转向(g)；

(g)计算鸟巢的适应值，并获得 $\hat{x}_b^{t+1}$，令 $t=t+1$，转(b)。

4 数值仿真与分析

为了验证所提算法的有效性，设计了仿真实验。在此选

择 CS[4] 和 ICS[11] 两种算法与本文提出的算法(称为 MSCS)进行对比。实验在相同的 Window 7, Pentium Dual-Core CPU T4300@2.10 GHz RAM 2.00G, MATLAB 7 环境下进行,并且选取文献[11,16]中的5个测试函数,具体信息见表1。

表1 测试函数名称、维数、搜索区间及最优值

函数名	维数	搜索区间	最优值
Sphere(f_1)	n	$[-100,100]^n$	0
Rosenbrock(f_2)	n	$[-10,10]^n$	0
Schwefel P2.22(f_3)	n	$[-100,100]^n$	0
Rastrigrin(f_4)	n	$[-100,100]^n$	0
Griewank(f_5)	n	$[-600,600]^n$	0

5个测试函数特点如下:f_1,f_2 和 f_3 是单峰函数,其中 f_2 的最优点被狭长的斜坡环绕,找到最优点比较困难;f_4 和 f_5 是多峰函数,并且 f_4 的尖峰个数随问题维数的增加而增加,f_5 存在多个局部最优点,因此这两个函数都有一定的求解难度。

参数选取:对于 CS 算法,采用文献[4]的参数,即 $P_a=0.25$,$\alpha=1$;对于 ICS 算法,取文献[11]中的参数,即 $P_a(\max)=1$,$P_a(\min)=0.005$,$\alpha(\max)=0.5$,$\alpha(\min)=0.05$,两种算法初始化鸟巢的数目取 $N=30$;对于 MSCS 算法,为了保证评价的客观性和公平性,我们也取 $N=30$,$P_a=0.25$,对于 MSCS 算法中其余参数 P_0,β_0,β_1 及 CR,虽然它们的不同组合能够得到较好结果,但是取 $P_0=0.5$,$\beta_0=0.3$,$\beta_1=0.6$,$CR=0.4$ 是最好的结果。3种算法的终止条件为满足最大迭代步1000或精度到达 10^{-5}。针对每一种算法,程序独立运行30次,获取统计数据与相应的仿真曲线,以便进行比较。表2—表6分别给出3种算法对测试函数 f_1-f_5 在维数 $n=10$,30,50,100下的最好值、最差值、平均值及达到精度时所需的迭代步数,表中"—"表示达到最大迭代时,没有达到所给精度。

表2 3种算法对 f_1 的优化结果

维数	算法	最好值	最差值	平均值	迭代数
10	CS	1.1347e−13	5.3551e−14	1.5495e−12	581
	ICS	3.7875e−29	3.3601e−027	2.0446e−28	292
	MSCS	**3.7862e−43**	**5.7425e−41**	**1.34274e−42**	**223**
30	CS	1.2421e−6	7.3248e−6	4.7492e−06	959
	ICS	2.9807e−12	4.1015e−10	3.6131e−11	597
	MSCS	**3.4238e−15**	**1.7647e−13**	**3.6092e−14**	**461**
50	CS	41.0909	77.9476	65.4582	—
	ICS	0.1256	0.2530	0.2019	—
	MSCS	**1.3018e−03**	**2.9233e−03**	**2.4264e−03**	—
100	CS	9.8506e+03	1.3883e+04	9.9553e+03	—
	ICS	1.5061e+03	1.3324e+03	1.2777e+03	—
	MSCS	**67.2011**	**1.3881e+02**	**1.0780e+02**	—

表3 3种算法对 f_2 的优化结果

维数	算法	最好值	最差值	平均值	迭代数
10	CS	0.6114	1.3829	0.8576	—
	ICS	0.1137	0.7236	0.0444	—
	MSCS	**1.1272e−07**	**3.8011e−07**	**1.7343e−07**	**851**
30	CS	2.5120e+01	2.6035e+01	2.5431e+01	—
	ICS	2.3901e+01	2.4826e+01	2.4392e+01	—
	MSCS	**1.1256e+01**	**2.2353e+01**	**2.0779e+01**	—
50	CS	48.1037	72.6935	55.2758	—
	ICS	48.2470	55.0389	52.7923	—
	MSCS	**44.9320**	**45.4698**	**45.1062**	—
100	CS	2.9028e+02	3.7573e+02	3.4354e+02	—
	ICS	2.5429e+02	3.3190e+02	3.1601e+02	—
	MSCS	**1.1216e+02**	**1.8530e+02**	**1.6384e+02**	—

表4 3种算法对 f_3 的优化结果

维数	算法	最好值	最差值	平均值	迭代数
10	CS	7.9006e−08	1.4779e−07	1.3801e−07	774
	ICS	8.8762e−18	1.2835e−17	3.0969e−18	283
	MSCS	**4.2371e−25**	**3.3779e−24**	**3.2428e−24**	**243**
30	CS	5.2037e−02	6.1002e−01	3.4243e−01	—
	ICS	9.0238e−09	5.7846e−08	1.8501e−08	746
	MSCS	**1.1939e−12**	**3.2479e−10**	**1.8614e−10**	**676**
50	CS	0.2144	0.2921	0.2522	—
	ICS	0.0103	0.0161	0.0128	—
	MSCS	**6.9539e−04**	**1.3507e−03**	**7.7422e−04**	—
100	CS	5.9344	6.5391	6.3018	—
	ICS	7.2664	17.3603	11.9748	—
	MSCS	**0.3433**	**0.6101**	**0.4021**	—

表5 3种算法对 f_4 的优化结果

维数	算法	最好值	最差值	平均值	迭代数
10	CS	5.5774e−11	5.5315e−09	2.9514e−10	661
	ICS	0	0	0	236
	MSCS	0	0	0	**184**
30	CS	1.4660e−02	9.8736e−01	2.2637e−02	—
	ICS	2.1208e−06	2.8235e−04	1.1084e−05	—
	MSCS	**5.9923e−08**	**8.0423e−07**	**6.8221e−07**	**849**
50	CS	2.5886	13.6641	4.2065	—
	ICS	3.3915e−03	1.0764e−02	4.4180e−03	—
	MSCS	**2.1706e−05**	**2.4521e−04**	**2.6713e−05**	—
100	CS	14.0831	30.1582	15.7740	—
	ICS	12.4125	16.3091	13.1353	—
	MSCS	**0.1530**	**0.9513**	**0.2938**	—

表6 3种算法对 f_5 的优化结果

维数	算法	最好值	最差值	平均值	迭代数
10	CS	0.0260	0.0548	0.0294	—
	ICS	0.0173	0.0480	0.0427	—
	MSCS	**1.0013e−04**	**0.0220**	**0.0141**	—
30	CS	4.5245e−03	1.0775e−02	7.7371e−02	—
	ICS	1.6509e−05	8.2834e−05	2.3295e−05	—
	MSCS	**2.1705e−10**	**1.9974e−08**	**4.2398e−09**	**685**
50	CS	0.7468	0.9935	0.9791	—
	ICS	0.0139	0.0213	0.1756	—
	MSCS	**1.0722e−04**	**4.6892e−04**	**2.5706e−04**	—
100	CS	3.0451	3.6999	3.2468	—
	ICS	1.1743	1.3112	1.2511	—
	MSCS	**0.7836**	**0.9576**	**0.8275**	—

从表2—表6可以看出,当 $n=10$,对于函数 f_1,f_2,f_3,在寻优能力及迭代步上,MSCS 算法优于 CS 和 ICS 两种算法;对于函数 f_4,MSCS 在寻优能力上优于 CS 算法,与 ICS 算法一样,但在迭代步上优于 CS 和 ICS 两种算法;对于函数 f_5,虽然3种算法都没有达到所要求的精度,但 MSCS 算法计算结果好于其它两种算法。当 $n=30$,对于函数 f_1,f_3,f_4 和 f_5,在寻优能力及迭代步上,MSCS 算法优于 CS 和 ICS 两种算法;对于函数 f_2,3种算法寻优能力相当。当 $n=50$ 和 $n=100$,对所有函数,3种算法都没有达到所要求的精度,但是 MSCS 算法获得的最好值、最差值和平均值明显优于其它两种算法。

为了反映算法的收敛速度,图1—图5分别给出了3种算法在维数 $n=30$ 下的5个函数收敛曲线。从图1、图4和图5可以看出,MSCS 收敛最快,CS 次之,ICS 最差。从图2和图3可以看出,MSCS 收敛最快,ICS 次之,CS 最差。具体说,对于函数 f_1 和 f_5,MSCS 算法收敛速度明显优于其它两种算法;对于函数 f_2 和 f_3,MSCS 算法收敛速度与 ICS 算法相比优势不大,但明显优于 CS 算法;对于函数 f_4,MSCS 算法与

其它两种算法相比优势不太明显，但好于其它两种算法。总之，MSCS最优值下降速度比CS和ICS快，这说明MSCS加速了优化过程。仿真结果表明，本文的MSCS算法不仅提高了搜索精度，而且加速了布谷鸟算法的收敛。

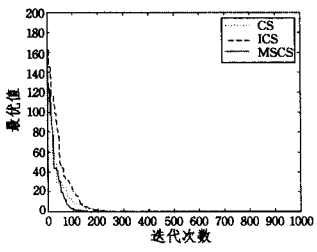

图1 f_1 收敛曲线

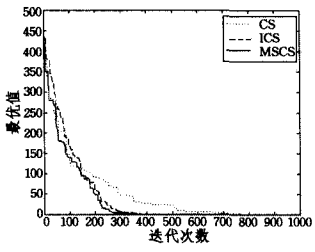

图2 f_2 收敛曲线

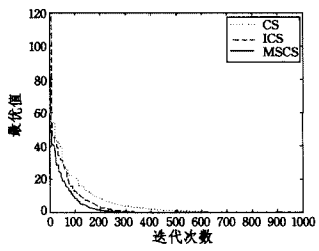

图3 f_3 收敛曲线

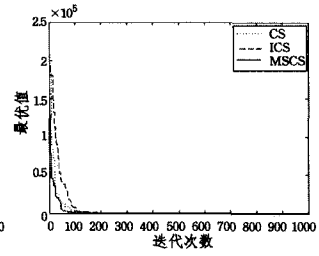

图4 f_4 收敛曲线

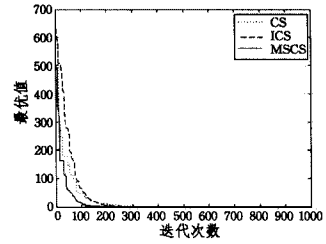

图5 f_5 收敛曲线

结束语 布谷鸟算法借助于自然界中布谷鸟繁衍后代的特性，融入“莱维飞行”这一特殊机制，从而实现了优化求解的能力。本文在原有的CS算法中引入一种参数自适应控制方法，并添加了一种新的变异机制，用于增强种群的多样性，旨在提高CS的探索能力、求解精度和收敛速度。仿真结果表明，所提出的MSCS算法是行之有效的，并具有更好的优化性能。

参考文献

[1] 朱钰，韩昌佩. 一种种群自适应收敛的快速遗传算法[J]. 计算机科学，2012，39(10)：214-217

[2] 李朔枫，李太勇. 一种基于距离的自适应模糊粒子群优化算法[J]. 计算机科学，2011，38(8)：257-259

[3] 傅嗣鹏，乔俊飞，韩红桂. 基于锦标赛选择变异策略的改进差分进化算法及函数优化[J]. 计算机科学，2013，40(6A)：15-18

[4] Yang X S, Deb S. Cuckoo Search via Lévy flights [C]// Proc. World Congress on Nature & Biologically Inspired Computing. IEEE Publications, India, 2009: 210-214

[5] Gandomi A H, Yang X S, Alavi A H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems [J]. Engineering with Computers, 2013, 29: 17-35

[6] Yang X S, Deb S. Engineering Optimization by Cuckoo Search [J]. Int. J. of Mathematical Modeling and Numerical Optimization, 2010, 1(4): 330-343

[7] Yildiz A R. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations [J]. International Journal of Advanced Manufacturing Technology, 2013, 64: 55-61

[8] 高述涛. CS算法优化神经网络的短时交通流量预测[J]. 计算机工程与应用，2013，49(9)：106-109

[9] Layeb A. A novel quantum inspired cuckoo search for Knapsack problems [J]. International Journal of Bio-inspired Computation, 2011, 3(5): 297-305

[10] Walton S, Hassan O, Morgan K, et al. Modified cuckoo search: A new gradient free optimization algorithm [J]. Chaos, Solitons & Fractals, 2011, 44(9): 710-718

[11] Valian E, Mohanna S, Tavakoli S. Improved Cuckoo Search Algorithm for Global Optimization [J]. Int. J. Communications and Information Technology, 2011, 1(1): 31-44

[12] 郑洪清，周永权. 一种自适应步长布谷鸟搜索算法[J]. 计算机工程与应用，2013，49(10)：68-71

[13] 王利英，杨绍普，赵卫国. 基于改进布谷鸟搜索算法的架桥机结构损伤识别[J]. 北京交通大学学报，2013，37(4)：168-173

[14] Zheng H Q, Zhou Y Q. A Novel Cuckoo Search Optimization Algorithm Base on Gauss Distribution [J]. Journal of Computational Information Systems, 2012, 8(10): 4193-4200

[15] 杜利敏，阮奇，冯登科. 基于共轭梯度的布谷鸟搜索算法[J]. 计算机与应用化学，2013，30(4)：406-410

[16] Ahmed S T, Amr A B, Ibrahim F A R. One rank cuckoo search algorithm with application to algorithmic trading systems optimization[J]. International Journal of Computer Applications, 2013, 64(6): 30-37

[17] Zheng H Q, Zhou Y Q, Gao P. Hybrid genetic-cuckoo search algorithm for solving runway dependent aircraft landing problem [J]. Research Journal of Applied Science, Engineering and Technology, 2013, 6(12): 2136-2140

[18] Yang X S, Deb S. Multiobjective cuckoo search for design optimization [J]. Computer' Operations Research, 2013, 40(6): 1616-1624

[19] Burnwal S, Deb S. Scheduling optimization of flexible manufacturing system using cuckoo search-based approach [J]. The International Journal of Advanced Manufacturing Technology, 2013, 64: 951-959

[20] Ma J M, Ting T O, Zhang N. Parameter Estimation of Photovoltaic Models via Cuckoo Search [J]. Journal of Applied Mathematics, 2013, 2013

[21] Civicioglu P, Besdok. A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms [J]. Artificial Intelligence Review, 2013, 39(4): 315-346