第38卷 第12期 计 算 机 工 程 2012年6月
Vol.38 No.12 Computer Engineering June 2012

·人工智能及识别技术· 文章编号：1000—3428(2012)12—0132—04 文献标识码：A 中图分类号：TP18

一种自适应惯性权重的混合蛙跳算法

刘悦婷[1]，赵小强[2]

(1. 甘肃联合大学电子信息工程学院，兰州 730000；2. 兰州理工大学电气工程与信息工程学院，兰州 730050)

摘 要：针对混合蛙跳算法(SFLA)易陷入局部最优、收敛速度慢的问题，提出一种改进的混合蛙跳算法。该算法用相对基学习法初始化青蛙群体，从而提高初始解的质量。通过引入自适应惯性权重修正青蛙的更新策略，可以平衡算法的全局搜索和局部搜索。对6个经典函数的仿真测试结果表明，该算法与SFLA和ISFLA1算法相比寻优能力强、迭代次数少、解的精度高，更适合高维复杂函数的优化。

关键词：混合蛙跳算法；相对基学习法；惯性权重；自适应；更新策略；全局最优

Adaptive Inertia Weight Shuffled Frog Leaping Algorithm

LIU Yue-ting[1], ZHAO Xiao-qiang[2]

(1. School of Electronics and Information Engineering, Gansu Lianhe University, Lanzhou 730000, China;
2. College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou 730050, China)

【Abstract】 Because of the problems of Shuffled Frog Leaping Algorithm(SFLA) such as local optimality and slow convergence rate, an improved SFLA is presented. In this algorithm, frog population is initialized with opposition base learning to improve the quality of initial solution. Then the adaptive inertia weight is introduced to correct frog update strategy which can balance the global search and local search. Simulation results of experiments on the six classical function show that compared with the SFLA and ISFLA1, the new algorithm optimization ability can be stronger, the number of iterations less, the solution better, the suitable for high-dimensional optimization of complex functions more.

【Key words】 Shuffled Frog Leaping Algorithm(SFLA); opposition base learning; inertia weight; adaptive; update strategy; global optimum

DOI: 10.3969/j.issn.1000-3428.2012.12.039

1 概述

混合蛙跳算法是一种受自然生物模仿启示而产生的、基于群体的协同搜索方法[1]。该算法模拟青蛙群体寻找食物时，按种群分类进行思想传递的过程，将全局广泛搜索和局部深度搜索相结合，使算法向着全局最优解的方向进行[2]。局部搜索使得子群内青蛙的个体信息得到交流，全局搜索使得整个群体青蛙间能够交流。与其他优化算法相比，SFLA具有概念简单、参数少、易于编程实现，并且结合广泛全局搜索和深度局部搜索的优势，因而既适合理论研究，又适合工程应用。

作为重要的优化工具，SFLA已广泛应用于资源网络优化问题、连续优化问题、聚类问题、流水车间调度问题等领域。而如何提高收敛速度和易陷入局部最优也成为该算法的一个研究热点。文献[3]为提高SFLA的收敛效率，提出在SFLA深度搜索方向上融合极值动力学优化算法。文献[4]针对SFLA易陷入局部最优，求解精度低的缺点，提出利用差分进化中的变异思想对SFLA的更新策略进行局部扰动。文献[5]在全局搜索中加入自适应logistic序列的混沌变异操作，同时结合个体适应度和进化代数自适应调整变异尺度，从而提高算法求解最优解的能力。文献[6]引入遗传算子增加对局部极值的扰动以避免陷入局部最优，同时对青蛙移动策略进行了优化，从而提高算法的性能和解的质量。受此启发，本文利用相对基学习方法产生初始群体，根据适应度值选出较优的群体，以提高初始解的质量，从而有助于提高最优解的质量。再引入自适应惯性权重对青蛙的更新策略进行优化。

2 SFLA介绍

对于d维问题，随机生成F只青蛙(解)组成初始群体，第i只青蛙可以表示为$X_i=(x_{i1},x_{i2},\cdots,x_{id})$，将种群内青蛙个体按适应度降序排列。然后将整个群体分为s个子群，每个子群包含n只青蛙，满足$F=s\times n$。第1只青蛙被分入第1个子群，第2只青蛙被分入第2个子群，…，第s只青蛙被分入第s个子群，第$s+1$只青蛙被分入第1个子群。依次类推，直到所有青蛙分配完毕[7]。

在每个子群中，具有最好适应度的青蛙记为X_b，最差适应度的青蛙记为X_w，而整个群体中具有最好适应度的青蛙记为X_g，对每个子群进行局部搜索，每次迭代只更新子群中的最差青蛙，更新策略为：

$$D_i=rand()\cdot(X_b-X_w) \tag{1}$$

$$X_w=X_w(\text{当前位置})+D_i\ (-D_{\max}\le D_i\le D_{\max}) \tag{2}$$

其中，D_i表示分量i上移动的距离；$rand()$是0~1之间的随机数；$D_{\max}$是青蛙允许移动的最大距离。

经过更新后，如果得到的解X_w优于原来的解X_w(当前位置)，则取代原来种群中的解；否则用X_g取代X_b，重复执行更新策略式(1)、式(2)；如果仍然没有改进，则随机产生一个新解取代原来的X_w。重复这种更新操作，直到设定的迭代次数，就完成一轮各子群的局部搜索。然后将所有子群的青蛙重新混合排序，划分子群，进行下一轮的局部搜索，如此反复直到满足终止条件。

基金项目：甘肃省支撑计划基金资助项目(090GKCA034)；甘肃省自然科学基金资助项目(0916RJZA017)

作者简介：刘悦婷(1979－)，女，讲师、硕士研究生，主研方向：电子自动控制；赵小强，副教授、博士

收稿日期：2011-07-21 **E-mail：**liuyueting996@qq.com

3　自适应惯性权重的混合蛙跳算法

3.1　初始群体的生成

初始群体的分布性质严重影响整个算法的收敛性能。若初始群体性质差，不但会造成算法收敛速度慢，而且会使算法不收敛。在标准 SFLA 中，随机生成的初始群体的各青蛙都集中在解空间的某一局部区域，对算法的性能有很大的影响。因此，本文利用相对基学习初始化青蛙群体，它是一种机器学习算法[8-9]。其主要思想是：在求解优化问题时，考察一个可行解的同时又考察它的相对解，通过评价这 2 个解优劣的来获得较优的候选解。这种同步机制为寻优提供了一个机会，因而可通过减少收敛所需的迭代次数来减少算法的运行时间，具体实现如下：

随机生成均匀分布的初始群体 $Y=\{Y_i \mid i=1,2,\cdots,F\}$ ；

计算相对群体 OY ，分别对 Y 中每只青蛙按式(3)计算其相对青蛙的位置，构成相对群体 $OY=\{OY_i \mid i=1,2,\cdots,F\}$ ；

$$oy_{id}=L_d+U_d-y_{id}(i=1,2,\cdots,F;\ d=1,2,\cdots,D) \tag{3}$$

其中， F 为青蛙群体的总数； D 为解的维数； U_d 、 L_d 分别表示第 d 维分量取值的上、下界($y_{id}\in[L_d,U_d]$)。

根据青蛙个体适应度值从 Y 和 OY 中选取 F 只作为初始种群：

$$Y^0=\{Y_i^0 \mid i=1,2,\cdots,F\} \tag{4}$$

满足：

$$Y_i^0=\begin{cases}Y_i & Y_i\text{优于}OY_i\\ OY_i & \text{其他}\end{cases}$$

3.2　改进的更新策略

为了充分体现 SFLA 群体智能行为的信息共享机制，提高局部搜索效率，借鉴 PSO 算法中离子更新策略，对 SFLA 中最差青蛙的移动步长的更新策略进行改进。

借鉴文献[10]，通过在 SFLA 中引入自适应惯性权重 ω 来协调该算法的全局和局部搜索能力。具体实现是将式(1)的移动步长计算公式变为式(5)，同时式(2)的位置更新保持不变。

$$D_i=\omega D_i^{'}+rand()\cdot(X_b-X_w) \tag{5}$$

其中， $D_i^{'}$ 为前一个最差青蛙的移动步长； ω 为惯性权重。

3.3　ω 的自适应调整

惯性权重 ω 表示青蛙运动的趋势，在局部最优解附近， ω 应取得较大，有利于跳出局部最优解，避免早熟；在全局最优解附近， ω 应取得较小，有利于在全局最优解附近更精细地搜索，更迅速地找到最优解。据此 ω 按式(6)进行自适应调整。

$\omega_i(t)$ 表示第 t 次第 i 只青蛙的惯性权重； $f_i(t)$ 表示第 t 次第 i 只青蛙的自适应度值； $\omega_{\min}$ 、 $\omega_{\max}$ 分别表示惯性权重最小值和最大值； $f_{\min}(t)$ 、 $f_{\max}(t)$ 、 $f_{\text{avg}}(t)$ 分别表示第 t 次整个群体自适应度值的最小值、最大值和平均值，则有：

$$\omega_i(t)=\begin{cases}\omega_{\min}+a(\omega_{\max}-\omega_{\min}) & f_i(t)<f_{\text{avg}}(t)\\ \omega_{\max}+b(\omega_{\max}-\omega_{\min}) & f_i(t)\geq f_{\text{avg}}(t)\end{cases} \tag{6}$$

其中， $a=(f_i(t)-f_{\min}(t))/(f_{\text{avg}}(t)-f_{\min}(t))$ ； $b=(f_i(t)-f_{\max}(t))/(f_{\text{avg}}(t)-f_{\max}(t))$ ； $t=it_j\times tit_k$ ， it_j 为当前子群内的迭代次数， it 为子群内迭代总次数， tit_k 为当前混合迭代次数， tit 为混合迭代总次数， $j\in[1,it]$ 的正整数， $k\in[1,tit]$ 的正整数。

3.4　新算法的流程

新算法的流程如下：

Step1 参数初始化，青蛙种群数 F ，子群数 s ，每个子群内青蛙数 n ，子群内迭代总数 it ，混合迭代总数 tit 。

Step2 利用式(3)、式(4)相对基学习法初始化群体。

Step3 对每只青蛙计算其自适应度值 $f(x_i)$ (即目标函数值)，按照 $f(x_i)$ 以降序对青蛙群体进行排序并分为 s 个子群。

Step4 确定每个子群中的 X_b 、 X_w 及群体全局最优 X_g ，对每个子群中最差青蛙按式(6)自适应调整 ω ，再按式(5)、式(2)进行更新，直到设定的迭代总次数 it ，对更新后的子群进行混合，取代原来的群体。

Step5 若当前迭代次数达到预定设定的最大次数 tit ，则迭代停止，输出最优适应度值的相关信息，算法结束，否则转到 Step3。

4　仿真实验

为了验证新算法的性能，选取 6 个经典的连续函数[10]进行测试，并与标准混合蛙跳算法 SFLA、ISFLA1[10]进行比较，6 个测试函数为：

(1)Sphere 简单的单峰函数

$$f_1(x)=\sum_{i=1}^{n}x_i^2,\ -100\leq x\leq 100$$

(2)Rosebrock 简单的多峰函数

$$f_2(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2],\ -30\leq x\leq 30$$

(3)Rastrigin 复杂的多峰函数

$$f_3(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10],\ -5.12\leq x\leq 5.12$$

(4)Griewank 复杂的多峰函数

$$f_4(x)=\frac{1}{4\ 000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1,\ -600\leq x\leq 600$$

(5)Ackley 复杂的多峰函数

$$f_5(x)=-20\exp(-0.2\sqrt{1/n\sum_{i=1}^{n}x_i^2})-\exp(1/n\sum_{i=1}^{n}\cos 2\pi x_i)+20+1,\ -32\leq x\leq 32$$

(6)Schafferf7 复杂的多峰函数

$$f_6(x)=\sum_{i=1}^{n-1}(x_i^2+x_{i+1}^2)^{0.25}[\sin^2(50(x_i^2+x_{i+1}^2)^{0.1})+1],\ -100\leq x\leq 100$$

在算法性能测试中，采用文献[11]中建议的参数设置，青蛙总数 F =200，子群数 s =20，子群内迭代总次数 it =10，混合迭代总次数 tit =500。

4.1　不同维函数的测试

函数变量维数 D =30，每种算法运行 30 次，3 种算法对 6 个测试函数的实验结果如表 1~表 6 所示。

表 1　Sphere 函数的仿真结果

算法	维数	平均最优适应度值	最优适应度值	标准差	CPU 运行时间/s
SFLA	2	0.018 316	0.016 263	0.009 214	0.9
	30	0.012 404	0.011 481	0.010 513	9.8
ISFLA1	2	0.002 879	0.002 598	0.000 026	1.2
	30	0.002 670	0.002 204	0.000 054	25.4
新算法	2	0.002 124	0.002 037	0.000 051	1.1
	30	0.002 035	0.001 984	0.000 049	24.6

表 2　Rosebrock 函数的仿真结果

算法	维数	平均最优适应度值	最优适应度值	标准差	CPU 运行时间/s
SFLA	2	45.316 842	40.302 689	12.166 327	1.9
	30	213.612 263	178.023 085	74.344 823	10.7
ISFLA1	2	8.493 211	4.915 394	4.131 521	2.1
	30	45.985 116	34.253 711	27.263 672	22.4
新算法	2	5.321 101	3.025 841	2.110 231	1.9
	30	12.021 311	7.243 682	6.121 325	19.3

表 3 Rastrigin 函数的仿真结果

算法	维数	平均最优适应度值	最优适应度值	标准差	CPU 运行时间/s
SFLA	2	4.123 614	2.289 641	1.021 103	1.5
	30	17.459 959	10.582 341	7.570 295	10.2
ISFLA1	2	2.310 211	1.504 852	0.810 113	1.8
	30	10.229 316	7.358 214	4.332 496	25.8
新算法	2	2.018 247	1.118 247	0.613 716	1.6
	30	7.211 736	5.014 537	2.946 521	24.7

表 4 Griewank 函数的仿真结果

算法	维数	平均最优适应度值	最优适应度值	标准差	CPU 运行时间/s
SFLA	2	0.923 612	0.613 451	0.231 014	0.9
	30	0.867 100	0.571 132	0.211 202	9.3
ISFLA1	2	0.141 357	0.099 786	0.077 139	1.7
	30	0.153 619	0.094 325	0.080 542	26.2
新算法	2	0.132 103	0.090 011	0.071 211	1.4
	30	0.091 299	0.050 121	0.079 272	25.1

表 5 Ackley 函数的仿真结果

算法	维数	平均最优适应度值	最优适应度值	标准差	CPU 运行时间/s
SFLA	2	1.594 532	0.748 952	0.735 642	1.2
	30	1.534 676	0.657 951	0.727 352	9.9
ISFLA1	2	0.039 458	0.036 138	0.003 692	1.9
	30	0.042 733	0.039 476	0.004 903	30.5
新算法	2	0.048 961	0.039 846	0.004 128	1.7
	30	0.031 546	0.030 946	0.004 652	28.1

表 6 Schafferf7 函数的仿真结果

算法	维数	平均最优适应度值	最优适应度值	标准差	CPU 运行时间/s
SFLA	2	25.126 891	9.235 648	6.852 143	2.1
	30	28.801 496	12.568 213	7.090 330	14.7
ISFLA1	2	16.452 687	8.142 638	6.658 234	2.6
	30	19.372 518	11.564 582	7.815 644	31.4
新算法	2	14.263 481	7.2534 67	6.245 311	2.4
	30	17.984 365	10.2648 32	6.984 513	29.8

分别比较 6 个表中的计算结果可知，在相同迭代次数及相同精度要求的前提下，新算法得到的平均最优适应度优于其他 2 种算法，并且标准差和 CPU 平均运行时间较小。由此可知，新算法与 SFLA、ISFLA1 相比有较好的全局搜索能力和较高的收敛精度，有效避免了早熟现象。可见本文在 SFLA 中引入相对基学习法和自适应调整惯性权重是可行的。

取测试函数维数为 30，预设精度为 10^{-9}，若算法在 500 次迭代中，最终结果满足精度要求，则该次运算算法收敛，反之为不收敛。算法收敛率为收敛次数与函数运行次数的比值。称算法收敛时，满足预设精度的迭代次数的最小值为算法的收敛代数，若最终优化结果不满足预设精度的要求，则该次收敛代数为指定迭代次数。平均收敛代数为各次收敛代数之和与函数运行次数的比值，实验结果如表 7 所示。

表 7 算法收敛性仿真结果

函数	仿真项	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
SFLA	收敛率/(%)	49	21	0	0	0	0
	平均收敛代数	381	450	500	500	500	500
ISFLA1	收敛率/(%)	60	47	21	15	0	0
	平均收敛代数	324	420	488	496	500	500
新算法	收敛率/(%)	100	100	100	100	100	100
	平均收敛代数	140	82	235	295	341	367

4.2 新算法中 ω 值的变化

图 1 为新算法中惯性权重的变化曲线。从图 1 可以看出，在进化早期，算法的惯性权重较小，局部搜索能力很强；在进化中期，采用自适应调整 ω 平衡了局部和全局的搜索能力；在进化的后期，局部搜索能力加强，所以自适应调整 ω 使算法的性能得到很大的提高。

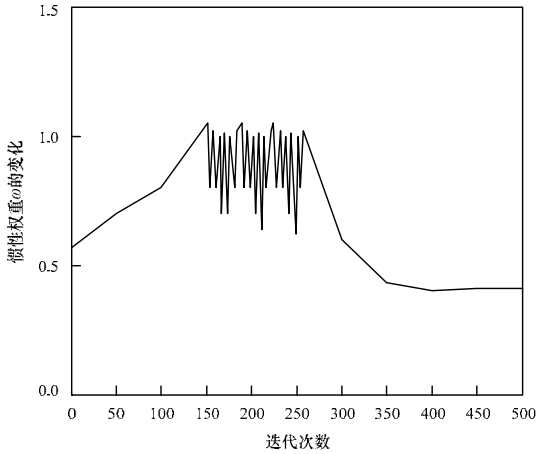

图 1 ω 随迭代次数的变化

4.3 高维函数的测试

为适应更大规模函数优化的目的，分别选取 Rosebrock、Ackley 和 Schafferf7 函数为测试对象，函数变量维数 D =150，3 种算法的迭代曲线如图 2~图 4 所示。考察 Rosebrock 函数从 30 维~200 维变化时，新算法的优化能力仿真曲线如图 5 所示。

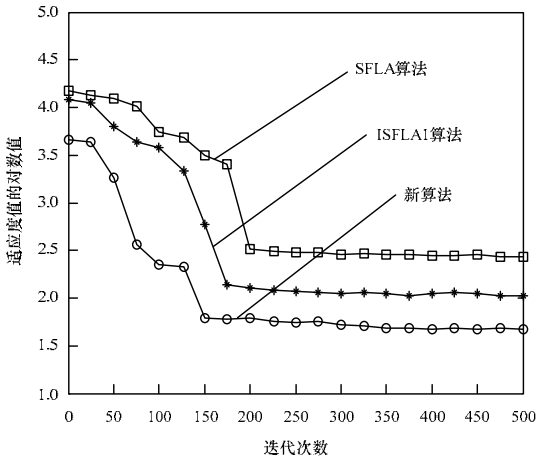

图 2 3 种算法对 Rosebrock 函数优化结果比较

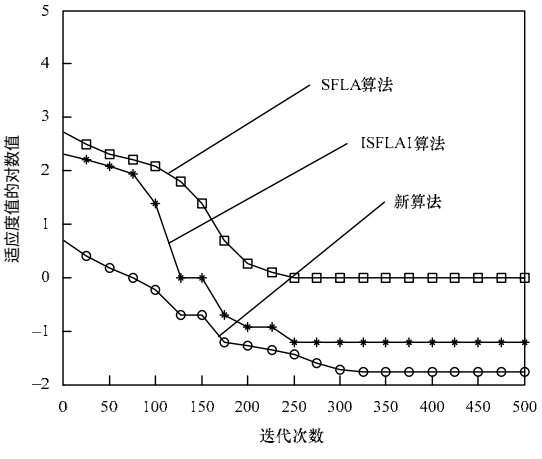

图 3 3 种算法对 Ackley 函数优化结果比较

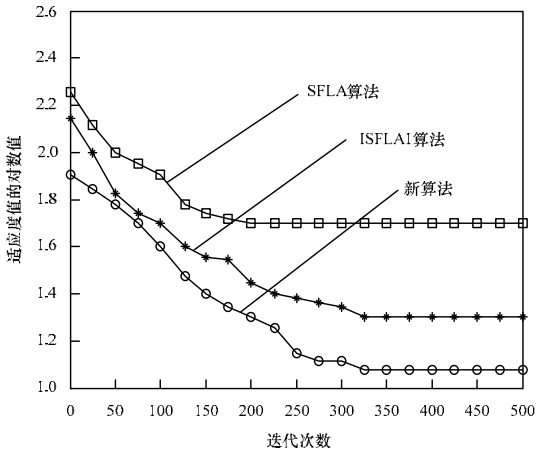

图 4　3 种算法对 Schafferf7 函数优化结果比较

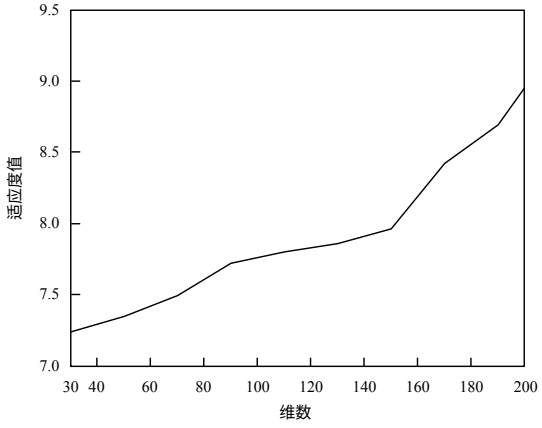

图 5　Rosebrock 函数优化结果随维数变化曲线

从图 2~图 4 可以看出，因为新算法采用相对基学习法初始化群体，所以它具有较好的初始解(初始适应度值较小)，而 SFLA 和 ISFLA1 初始群体随机分布，故初始解较差。由于新算法能自适应调整惯性权重，因此收敛速度加快，迭代次数较少，而 ISFLA1 中无惯性权重系数，所以在最优解附近徘徊时间较长，使得迭代次数较多，运行时间较长。

从图 5 可以看出，新算法的维数从 30 维到 150 维变化，相应的适应度值从 7.24 增加到 8.95，即目标函数的变化范围很小，表明新算法对高维函数仍有很强的优化能力。

5　结束语

本文针对混合蛙跳算法易陷入局部最优、收敛速度慢的问题，提出改进的蛙跳算法。该算法基于以下 3 个方面进行了改进：(1)用相对基学习法初始化青蛙群体，提高初始解的质量，从而减少迭代时间。(2)借鉴 PSO 算法更新机制，对 SFLA 的更新策略进行修正，提高局部搜索效率。(3)自适应调整惯性权重，根据实际情况选取合适的惯性权重，可以使算法在搜索精度和搜索速度方面都达到最优。实验结果表明，无论是二维还是高维问题，本文算法都得到了良好的效果，在高维情况下，该算法性能稳定，更适合求解高维复杂函数的优化问题。SFLA 的优化算法已取得了较多的研究成果，但参数设置和优化性能仍比较薄弱，这也是今后需要继续研究的方向。

参考文献

[1] Eusuff M M, Lansey K E. Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm[J]. Journal of Water Sources Planning and Management, 2003, 129(3): 210-225.

[2] 赵守法. 蛙跳算法的研究与应用[D]. 上海: 华东师范大学, 2008.

[3] 骆剑平, 陈泯融. 混合蛙跳算法及其改进算法的运动轨迹及收敛性分析[J]. 信号处理, 2010, 26(9): 1428-1433.

[4] 赵鹏军. 基于差分扰动的混合蛙跳算法[J]. 计算机应用, 2010, 30(10): 2575-2577.

[5] 葛　宇, 王学平, 梁　静. 自适应混沌变异蛙跳算法[J]. 计算机应用研究, 2011, 28(3): 945-947.

[6] 欧　阳, 孙元姝. 基于改进混合蛙跳算法的网格任务调度策略[J]. 计算机工程, 2011, 37(16): 1-3.

[7] Amiri B, Fathian M, Maroosi A. Application of Shuffled Frog-leaping Algorithm on Clustering[J]. The International Journal of Advanced Manufacturing Technology, 2009, 45(1/2): 199-209.

[8] Tizhoosh H R. Opposition-based Learning: A New Scheme for Machine Intelligence[C]//Proc. of International Conference on Computational Intelligence. Vienna, Austria: [s. n.], 2005: 695-701.

[9] Tizhoosh H R. Opposition-based Reinforcement Learning[J]. Journal of Advanced Computational Intelligence and Intelligent Informatics, 2006, 10(3): 578-585.

[10] 赵鹏军, 刘三阳. 求解复杂函数优化问题的混合蛙跳算法[J]. 计算机应用研究, 2009, 26(7): 2435-2437.

[11] Elbeltagi E, Hegazy T, Grierson D. Comparison Among Five Evolutionary-based Optimization Algorithms[J]. Advanced Engineering Informatics, 2005, 19(1): 43-53.

编辑　顾逸斐

(上接第 131 页)

参考文献

[1] 曹剑芬. 语音处理上如何逐渐减少对具体语料的依赖？[J]. 清华大学学报: 自然科学版, 2009, 49(S1): 1380-1387.

[2] 陶　梅. 基于 HTK 的维吾尔语连续语音识别研究[D]. 乌鲁木齐: 新疆大学, 2008.

[3] 冯丽娟, 吾守尔·斯拉木. 维吾尔语连续语音识别技术研究[J]. 现代计算机, 2010, (1): 4-7.

[4] 那斯尔江·吐尔逊, 吾守尔·斯拉木. 基于隐马尔科夫模型的维吾尔语连续语音识别系统[J]. 计算机应用, 2009, 29(7): 2009-2011.

[5] 伊·达瓦, 匂坂芳典, 中村哲. 语料资源缺乏的连续语音识别方法的研究[J]. 自动化学报, 2010, 36(4): 550-557.

[6] Julius[EB/OL]. [2008-05-13]. http://Julius.Sourceforge.jp/.

[7] 伊·达瓦, 大川茂树, 白井克彦. 蒙古语 7 个元音声学特性计算机分析[J]. 声学学报, 1999, 24(1): 94-97.

[8] Palmkit[EB/OL]. [1997-10-24]. http://palmkit.sourceforge.net/.

编辑　张正兴