对精英加速的改进人工鱼群算法*

李 君, 梁昔明
(北京建筑大学 理学院, 北京 102600)

摘 要: 人工鱼群算法是一种群智能全局随机优化算法,存在算法收敛精度低和效率差的缺点。为克服这一缺点,利用最速下降法具有运算简单、运算速度较快的特点,提出了对精英加速的改进人工鱼群算法。该算法利用最速下降法对适应度值最好的人工鱼进行更新,通过人工鱼之间信息交换指导其他人工鱼,提高鱼群整体水平,加快人工鱼群算法收敛速度。数值实验结果表明,所得改进人工鱼群算法不仅运算量减少,而且具有更快的收敛速度和更高的收敛精度。改进算法提高收敛精度和运算效率,相较其他算法具有一定的优势。

关键词: 人工鱼群算法; 最速下降法; 数值实验; 适应度函数
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2018)07-1960-05
doi:10.3969/j.issn.1001-3695.2018.07.008

Improved artificial fish swarm algorithm to accelerate elite

Li Jun, Liang Ximing
(*School of Science, Beijing University of Civil Engineering & Architecture, Beijing* 102600, *China*)

Abstract: Artificial fish swarm algorithm is a swarm intelligence global stochastic optimization algorithm, its disadvantages are poor local search capability and low efficiency. The steepest descent method has the advantages of simple operation and fast computation speed. This paper proposed an improved artificial fish swarm algorithm to accelerate the elite. This algorithm used steepest descent method to update artificial fish with the best fitness value, through exchange of information between fishes guide other artificial fishes, improved the overall level of fish swarm, accelerated the convergence rate of artificial fish swarm algorithm. Numerical test results show that, the improved artificial fish swarm algorithm not only reduces the computational complexity, but also has a faster convergence rate and a higher convergence precision. The improved algorithm improves the convergence precision and computational efficiency, and has some advantages compared with other algorithms.

Key words: artificial fish swarm algorithm(AFSA); steepest descent method; numerical experiment; fitness function

群智能优化算法已成为优化计算的重要研究方向。近年来一些新型随机搜索优化算法迅猛发展,其中较具代表性的有蚁群算法、粒子群算法和人工鱼群算法。人工鱼群算法(artificial fish swarm algorithm,AFSA)是李晓磊等人[1,2]通过研究鱼群的行为特点,并应用动物自治体的模型,于2002年首次提出的一种自下而上的新型寻优模式。鱼群算法是一种有效的寻优算法,具有良好的求取全局极值的能力,并具有对初值参数选择不敏感、鲁棒性强、简单易实现、寻优速度快等优点。但是随着优化问题复杂程度和规模的不断扩大,AFSA在应用中也存在着不足[3,4],主要表现在当寻优域较大或处于变化平坦的区域时,收敛于全局最优解的速度减慢且搜索性能劣化,算法一般在优化初期具有较快的收敛性,但后期往往收敛较慢、算法很难得到高精度的最优解、只比较容易找到满意的解的域、算法在计算复杂度较大的问题有很大缺陷等方面。因此,学者们进行了深入的研究,提出了一些改进算法,也都取得了不错的效果。其中,文献[5]在迭代中对当前种群中部分优质个体执行一般动态反向学习,生成它们的反向种群,引导种群向包含全局最优的解空间逼近的一种应用佳点集和反向学习的人工鱼群算法;文献[6]引入吞食行为和跳跃行为的改进人工鱼群算法;文献[7]针对传统 Ada Boost 人脸检测算法存在的不足,提出一种将人工鱼群算法、粒子群优化算法与 AdaBoost 算法相结合的人脸检测算法;文献[8~10]将人工鱼群算法与一些智能方法相结合,虽然都较好地改善了AFSA,使算法精度得到提高,但由于过分增加全局搜索能力或局部搜索能力,从而导致种群的迭代次数增加,增加了算法的计算量,降低了算法的效率。目前人工鱼群算法已在各方面得到了广泛应用[11,12],如金融信息[13]、神经网络[14]、路径问题[15]、煤炭行业[16]、多用户检测器[17]和非线性复杂函数最优化等方面,且都取得了较好的效果。本文提出了一种利用最速下降法对精英人工鱼加速的改进人工鱼群算法(LCAFSA),有效结合了基本人工鱼群算法较强的全局搜索能力和最速下降法快速的局部搜索能力。数值实验结果表明,所得改进的人工鱼群算法具有更快的收敛速度和更高的收敛精度。

1 基本人工鱼群算法与最速下降法分析

对无约束连续优化问题:

$$\min f(x) \quad x \in \mathrm{R}^d \tag{1}$$

若 $x^* \in \mathrm{R}^d$ 满足

$$f(x^*) \leqslant f(x) \quad \forall x \in \mathrm{R}^d \tag{2}$$

则称 x^* 为 $f(x)$ 在全空间 R^d 上的全局极小点。

1.1 AFSA 原理

人工鱼群算法模拟自然界中鱼的集群觅食行为,通过个体之间的协作以及自我游动使群体达到寻优的目的。鱼群算法主要利用人工鱼群的觅食、聚群和追尾三种算子进行更新。每条人工鱼探索当前所在的环境,选择执行一种行为算子,通过不断调整自己的位置,最终集结在食物密度较大的区域周围,取得全局极值。人工鱼是真实鱼的虚拟实体。人工鱼存在的环境指的是问题的解空间和其他同伴的状态,下一刻的行为取决于目前自身状态和周围环境的状态,并且它还通过自身活动来影响环境和其他同伴的活动,进而达到最终寻优的目的。

在人工鱼群算子模型中包括三种主要的算子,即聚群算子、追尾算子和觅食算子。这三种算子是算法的核心,并且决

收稿日期: 2017-02-27; **修回日期**: 2017-04-19 **基金项目**: 国家自然科学基金资助项目(61463009);北京市自然科学基金资助项目(4122022);中央支持地方科研创新团队项目(PXM2013-014210-000173)

作者简介:李君(1989-),男,硕士研究生,主要研究方向为最优化方法及其应用(lijun130181@163.com);梁昔明(1967-),男,教授,博导,博士,主要研究方向为最优化方法及其应用.

定算法的性能和最优解搜索的准确度。设 X_i 表示第 i 条人工鱼的位置向量;Y_i 表示第 i 条人工鱼的适应度值,$Y_i = f(X_i)$;δ 表示拥挤度因子;try_number 表示人工鱼在觅食算子中最大的试探次数。

a)觅食算子。人工鱼 i 按式(3)随机选择当前感知范围内的一个位置 X_j,若 $Y_i > Y_j$,则按式(4)向该位置前进一步进行位置更新,使得 X_i 到达一个新的位置 X_{next};反之,则重新随机选择位置,判断是否满足前进条件;这样反复尝试 try_number 次后,如果仍不满足前进条件,则按式(5)在感知范围内随机移动一步。

$$X_j = X_i + \text{rand} \times \text{visual} \tag{3}$$

$$X_{\text{next}} = X_i + \text{rand} \times \text{step} \times \frac{X_j - X_i}{\| X_j - X_i \|} \tag{4}$$

$$X_{\text{next}} = X_i + \text{rand} \times \text{step} \tag{5}$$

b)聚群算子。人工鱼 i 以自身位置 X_i 为中心,其感知范围内的人工鱼的数目为 nf,当 $nf \geqslant 1$,这些人工鱼形成如式(6)所示的集合 S_i,按式(7)计算 S_i 的中心位置 X_{center}。如果满足 $Y_{\text{center}} < Y_i$ 且 $Y_{\text{center}}/nf > \delta \times Y_i$,表明伙伴中心有很多食物且不太拥挤,则按式(8)朝该中心方向前进一步;否则,执行觅食算子。

$$S_i = \{ X_j \mid \| X_j - X_i \| \leqslant \text{visual} \} \tag{6}$$

$$X_{\text{center}} = \sum_{j=1}^{nf} X_j / nf \tag{7}$$

$$X_{\text{next}} = X_i + \text{rand} \times \text{step} \times \frac{X_{\text{center}} - X_i}{\| X_{\text{center}} - X_i \|} \tag{8}$$

c)追尾算子。人工鱼 i 根据自己的当前位置 X_i 搜索其感知范围内所有伙伴中适应度值最小的伙伴 $X_{\min}$,其适应度值为 $Y_{\min}$。如果 $Y_{\min} < Y_i$,就以 $X_{\min}$ 为中心搜索其感知范围内的人工鱼,数目为 nf,并且满足 $Y_{\min} < Y_i$ 且 $Y_{\min}/nf > \delta \times Y_i$,表明该位置较优且其周围不太拥挤,则按式(9)向着适应度值最小伙伴 $X_{\min}$ 的方向前进一步;否则,执行觅食算子。

$$X_{\text{next}} = X_i + \text{rand} \times \text{step} \times \frac{X_{\min} - X_i}{\| X_{\min} - X_i \|} \tag{9}$$

人工鱼群算法中每条人工鱼首先分别试探执行聚群和追尾两种算子,通过目标函数计算其适应度值是否得到改善,将适应度值改善较大的算子作为该条人工鱼的更新算子;若某条人工鱼执行聚群和追尾两种算子后,适应度值均没有改善,则执行觅食算子;若这条人工鱼在达到觅食算子的最大尝试次数以后,适应度值仍然没有改善,则在自己周围环境中随机游动到一个新的位置。所有人工鱼执行完上述操作后,最终人工鱼群集结在几个局部最优解的周围,且全局最优的极值区域周围一般能集结较多人工鱼[18]。

1.2　最速下降法

最速下降法[19]是求解无约束优化问题最简单和最古老的方法之一。最速下降法是用负梯度方向为搜索方向的,计算过程就是沿梯度下降的方向求解极小值(也可以沿梯度上升方向求解极大值)。梯度方向可以通过对函数求导得到,一般确定步长的方法是由线性搜索算法来确定。因为一般情况下,梯度向量为 0 说明是到了一个极值点,此时梯度的幅值也为 0。而采用梯度下降算法进行最优化求解时,算法迭代的终止条件是梯度向量的幅值接近 0 即可,可以设置这个非常小的常数阈值。最速下降法具体计算步骤如下:

a)选取初始点 $x_0 \in \mathrm{R}^n$,允许误差 $0 \leqslant \varepsilon \leqslant 1$,令 $k = 1$。

b)计算 $g_k = \nabla f(x_k)$。若 $\| g_k \| \leqslant \varepsilon$ 或 $k \leqslant N$(为最大迭代次数),停算,输出 x_k 作为近似最优解。

c)取方向 $d_k = -g_k$。

d)由线搜索技术确定步长因子 α_k。

e)令 $x_{k+1} = x_k + \alpha_k d_k$,$k = k + 1$,转到步骤 b)。

2　对精英加速的改进人工鱼群算法

最速下降法是求解多变量函数极值问题最早的一种方法,主要用于求解无约束优化问题。从给定初始点 X_0 出发,初始方向 P_0 为该点处负梯度方向。由于 P 是下降方向,寻找到的下一个点 X_{k+1} 一定比点 X_k 更优,不会出现无价值迭代,其具有快速精细的局部搜索能力。利用最速下降法对适应度值最好的人工鱼进行更新,提出对精英加速的改进人工鱼群算法(LCAFSA)。在基本人工鱼群算法中,找到适应度值最好的人工鱼,利用最速下降法进行更新。具体更新规则为:以 X 为初始点,X 处的负梯度方向 P_0 为搜索方向,根据式(11)对 X 进行更新,其中步长 t 根据式(12)确定求解。如果 X_1 满足给定的精度要求 eps = 1e - 5,则跳出最速下降法循环;否则,按式(13)(14)进行下一点的迭代,直到满足要求的精度或迭代次数。

$$P_0 = -g(X) \tag{10}$$

$$X_1 = X_0 + tP_0 \tag{11}$$

$$f(X_0 + tP_0) = \min_{t \geqslant 0} f(X_0 + tP_0) \tag{12}$$

$$X_{k+1} = X_k + tP_k \tag{13}$$

$$P_k = -g(X_k) \tag{14}$$

此时迭代求得的位置显然比初始点 X 更优,但要求的当前最好解需进行多次迭代。从整体效率来考虑,利用精度和迭代次数一起构成最速下降法的终止条件,即当最速下降法循环 $k = 5$ 次仍没有达到要求的精度时,就跳出最速下降法的循环,得到新位置 X。算法 LCAFSA 流程如图 1 所示。

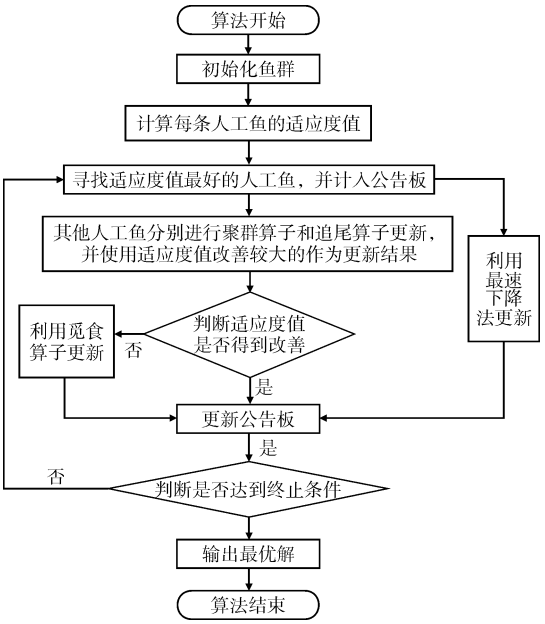

图 1　算法 LCAFSA 流程

改进的人工鱼群算法(LCAFSA)流程如下:

a)随机初始化 N 条人工鱼,给定视野 visual,步长 step,拥挤度因子 δ,最大重复次数 try_number,最大迭代次数 max,精度 tol。

b)计算每条人工鱼的适应度值,并寻找适应度值最好的人工鱼,记录在公告板。

c)对公告板内人工鱼利用最速下降法更新。

d)分别对除去适应度值最好的其余每条人工鱼进行聚群算子和追尾算子,选择适应度值改善较大的作为更新结果。

e)计算更新后人工鱼适应度值。若更新后适应度值没有得到改善,则利用觅食算子对该人工鱼进行更新。

f)将更新后人工鱼适应度值与公告板进行比较,若较好,则将其赋予公告板。

g)当所有人工鱼进行更新完成之后,判断是否达到终止条件(终止条件为达到最大迭代次数或达到目标精度)。如果满足终止条件,则输出最优解,算法终止;否则,转步骤 c)。

3　实验结果与分析

为了验证所提出的改进人工鱼群算法(LCAFSA)的性能,使用以下九个典型的无约束优化问题进行数值实验。各问题的目标函数表达式、搜索范围和理论最优值如下:

a) Six-hump camel back 函数。$f_1(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$,每个分量的搜索为[-5,5]。当 $x=(0.0898,-0.7126)$ 时,对应问题的全局最小值为 -1.031 6,此函数为多峰函数,有六个极值点。

b) Extended beale 函数。$f_2(x)=[1.5-x_1(1-x_2)]^2+[2.25-x_1(1-x_2^2)]^2+[2.625-x_1(1-x_2^3)]^2$,每个分量的搜索为[-5,5]。当 $x=(3,0.5)$ 时,对应问题的全局最小值为 0。

c) Shaffer 函数。$f_3(x)=(x_1+x_2)^{0.25}[\sin^2(50(x_1^2+x_2^2)^{0.1})+1]$,每个分量的搜索为[-100,100]。当 $x=(0,0)$ 时,对应问题的全局最小值为 0,此函数为复杂的多峰函数。

d) Ackley 函数。$f_4(x)=-20\exp(-0.2\sqrt{\sum_{i=1}^{n}x_i^2/n})-\exp(\sum_{i=1}^{n}\cos(2\pi x_i)/n)+20+e$,每个分量的搜索为[-32,32]。当 $x=(0,0)$ 时,对应问题的全局最小值为 0。

e) Sphere 函数。$f_5(x)=\sum_{i=1}^{n}x_i^2$,每个分量的搜索为[-100,100]。当 $x_i=0(i=1,2,\cdots,n)$ 时,对应问题的全局最小值为 0。

f) Griewank 函数。$f_6(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$,每个分量的搜索为[-600,600]。当 $x_i=0(i=1,2,\cdots,n)$ 时,对应问题的全局最小值为 0。

g) Rosenbrock 函数。$f_7(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)+(x_i-1)^2]$,每个分量的搜索为[-100,100]。当 $x_i=1(i=1,2,\cdots,n)$ 时,对应问题的全局最小值为 0。

h) Rastrigin 函数。$f_8(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$,每个分量的搜索为[-10,10]。当 $x_i=0(i=1,2,\cdots,n)$ 时,对应问题的全局最小值为 0。

i) Schwefel 函数。$f_9(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j^2)$,每个分量的搜索为[-100,100]。当 $x_i=0(i=1,2,\cdots,n)$ 时,对应问题的全局最小值为 0。

具体参数设置如下:人工鱼个体数 $N=20$;觅食算子中最大尝试次数 try_number = 5;对于目标函数 f_1、f_2、f_3,visual = 2.5,step = 0.5;对于目标函数 f_4,visual = 4.5,step = 3;对于目标函数 f_5、f_7、f_9,visual = 25,step = 3;对于目标函数 f_6,visual = 150,step = 18;对于目标函数 f_8,visual = 2.5,step = 0.3。目标函数 $f_1\sim f_4$ 取 2 维,目标函数 $f_5\sim f_9$ 取 10 维。

算法性能采用如下评价方法:a)固定迭代次数评价算法精度;b)固定收敛精度评估算法达到该精度所需迭代次数;c)与其他算法优化结果进行比较;d)调用目标函数次数和运行时间对比;e)10~30 维函数优化结果比较;f)分析参数对算法影响。

采用 MATLAB-R2014b 实验平台进行数值实验。为减少偶然性的影响,对每个测试问题进行 50 次独立实验取平均值。对目标函数 $f_1\sim f_9$,将算法 LCAFSA 与基本人工鱼群算法(AFSA)求得数据进行对比分析;文献[20]中只有测试函数 $f_5\sim f_9$ 的计算数据,因此对目标函数 $f_5\sim f_9$ 同时与文献[20]中的算法 SAFSA 进行对比。

3.1 固定迭代次数的收敛精度

图 2~10 为在固定迭代 1 000 次的情况下,算法 AFSA 和 LCAFSA 对目标函数 $f_1\sim f_9$ 平均解的进化曲线。

由图 2~10 可以看出,对目标函数 $f_1\sim f_9$,算法 AFSA 大概迭代 200 次前效果明显,但是之后效果不佳。由图 2 可以看出,对目标函数 f_1,算法 LCAFSA 与 AFSA 效果相差不大;由图 3 和 10 可以看出,对目标函数 f_2 和 f_9,算法 LCAFSA 快速收敛到较高精度;由图 4 可以看出,对目标函数 f_3,算法 LCAFSA 快速收敛到较高精度,直至找到最优解;由图 5 可以看出,对目标函数 f_4,算法 LCAFSA 前期与算法 AFSA 效果相差不大,但大概迭代 230 次时,快速找到高精度的解;由图 6 和 9 可以看出,对目标函数 f_5 和 f_8,算法 LCAFSA 快速地达到最优解;由图 7 可以看出,对目标函数 f_6,算法 LCAFSA 与 AFSA 迭代效果差不多,精度略微有所提升;由图 8 可以看出,对目标函数 f_7,算法 LCAFSA 后收敛速度略减,但仍迭代到较高精度。

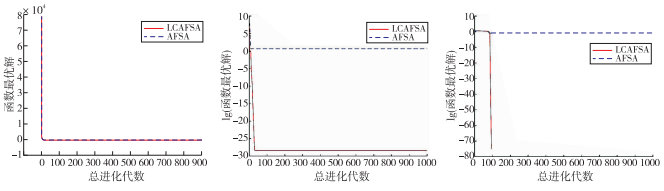

图 2 目标函数 f_1 平均解进化曲线　图 3 目标函数 f_2 平均解进化曲线　图 4 目标函数 f_3 平均解进化曲线

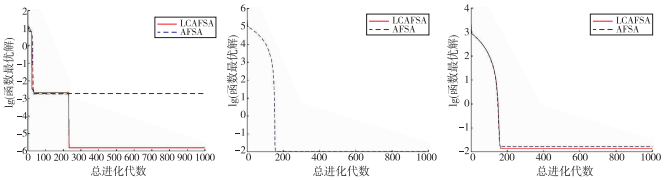

图 5 目标函数 f_4 平均解进化曲线　图 6 目标函数 f_5 平均解进化曲线　图 7 目标函数 f_6 平均解进化曲线

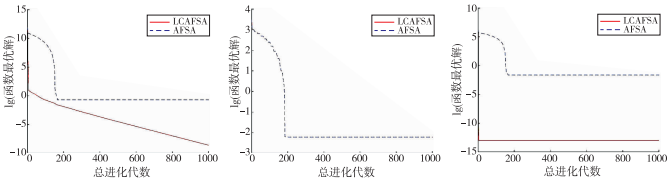

图 8 目标函数 f_7 平均解进化曲线　图 9 目标函数 f_8 平均解进化曲线　图 10 目标函数 f_9 平均解进化曲线

表 1 为算法 LCAFSA 与 AFSA 对目标函数 $f_1\sim f_4$ 迭代 1 000次的平均解、最好解和最差解的对比;表 2 为算法 LCAFSA、AFSA 和 SAFSA 对目标函数 $f_5\sim f_9$ 迭代 1 000 次的平均解、最好解和最差解的对比。表 2 中算法 AFSA 和 SAFSA 数据源自文献[20]。

表 1 在固定迭代次数下求解目标函数 $f_1\sim f_4$ 的结果对比

函数	算法	平均解	最好解	最差解
f_1	AFSA	-0.130 37	-0.260 76	1.37E-05
	LCAFSA	-1.031 6	-1.031 6	-1.031 6
f_2	AFSA	2.427 683	0.055 192	6.677 82
	LCAFSA	6.43E-27	1.04E-28	1.89E-26
f_3	AFSA	0.076 251	0.067 93	0.084 573
	LCAFSA	0	0	0
f_4	AFSA	2.81E-03	1.02E-03	7.86E-03
	LCAFSA	5.85E-06	3.21E-07	1.70E-06

表 2 在固定迭代次数下求解目标函数 $f_5\sim f_9$ 的结果对比

函数	算法	平均解	最好解	最差解
f_5	AFSA	0.549 313	0.190 388	1.707 638
	SAFSA	1.15E-08	5.04E-09	1.81E-08
	LCAFSA	0	0	0
f_6	AFSA	0.720 781	0.452 412	6.563 051
	SAFSA	0.118 239	0.000 766	0.477 634
	LCAFSA	1.54E-02	3.80E-04	0.0449834
f_7	AFSA	31 291.71	54.364 19	1 049 064
	SAFSA	4.098 545	0.883 168	7.822 047
	LCAFSA	2.55E-09	2.54E-09	2.62E-09
f_8	AFSA	137.325 5	86.727 03	183.595 2
	SAFSA	8.183 427	1.989 964	24.875 29
	LCAFSA	0	0	0
f_9	AFSA	847.150 8	4.123 204	5 343.838
	SAFSA	0.000 006	0.000 003	0.000 012
	LCAFSA	9.95E-14	9.95E-14	9.95E-14

从表 1 中可以看出,在迭代 1 000 次情况下,对目标函数 f_1 ~f_4,算法 AFSA 都得不到最优解;对目标函数 f_1 和 f_3,算法 LCAFSA 可以得到理论最优解;对目标函数 f_2,算法 LCAFSA 可以得到精度很高的解;对目标函数 f_4,算法 LCAFSA 的优化结果比算法 AFSA 有了一定提高。

从表 2 中可以看出,在迭代 1 000 次情况下,对目标函数 f_5 ~f_9,算法 AFSA 都得不到最优解;算法 SAFSA 虽然也得不到最优解,但是其解对精度有一定的提高;对目标函数 f_5 和 f_8,算法 LCAFSA 可以得到理论最优解;对目标函数 f_6,算法 LCAFSA 的优化结果比算法 AFSA 和 SAFSA 略高;对目标函数 f_7 和 f_9,算法 LCAFSA 的优化结果比算法 AFSA 和 SAFSA 的优化结果对精度有很大的提升,非常接近最优解。

由上可知,在固定迭代次数的情况下,算法 LCAFSA 的收敛精度高于算法 AFSA 和 SAFSA。

3.2　指定精度下的迭代次数

表 3 为算法 AFSA 和 LCAFSA 在指定精度下求解测试函数 f_1 ~f_4 迭代次数的对比;表 4 为算法 AFSA、SAFSA 和 LCAFSA 在指定精度下求解测试函数 f_5 ~f_9 迭代次数的对比。其中,成功率表示达到指定精度的运行次数除以独立运行次数 50。表 4 中算法 AFSA 和 SAFSA 数据源自文献[20]。目标函数 f_1 和 f_4 指定精度为 10^{-3};目标函数 f_2、f_3、f_5 和 f_9 指定精度为 10^{-5};目标函数 f_6 指定精度为 10^{-2};目标函数 f_7 和 f_8 指定精度为 10,最大运行次数为 1 000 次。

表 3　在指定精度下求解目标函数 f_1 ~f_4 的结果对比

函数	算法	成功率	平均迭代次数	最小迭代次数	最大迭代次数
f_1	AFSA	0			
	LCAFSA	100	9	4	14
f_2	AFSA	0			
	LCAFSA	100	38	5	56
f_3	AFSA	0			
	LCAFSA	100	64	61	67
f_4	AFSA	0			
	LCAFSA	100	224	57	402

表 4　在指定精度下求解目标函数 f_5 ~f_9 的结果对比

函数	算法	成功率	平均迭代次数	最小迭代次数	最大迭代次数
f_5	AFSA	0			
	SAFSA	100	606	589	630
	LCAFSA	100	2	1	2
f_6	AFSA	0			
	SAFSA	100	157	30	464
	LCAFSA	100	156	149	169
f_7	AFSA	0			
	SAFSA	98	421	365	557
	LCAFSA	100	5	3	5
f_8	AFSA	0			
	SAFSA	82	267	76	560
	LCAFSA	100	12	2	165
f_9	AFSA	0			
	SAFSA	100	631	617	647
	LCAFSA	100	2	2	2

从表 3 中可以看出,在指定收敛精度下,对目标函数 f_1 ~f_4,算法 AFSA 都没有得到达到精度的解,算法 LCAFSA 成功率均为 100%,并且迭代次数较少。

从表 4 中可以看出,在指定收敛精度下,对目标函数 f_5 ~f_9,算法 AFSA 都没有得到达到精度的解。对目标函数 f_5、f_6、f_9,算法 SAFSA 成功率为 100%;对目标函数 f_7,算法 SAFSA 成功率为 98%;对目标函数 f_8,算法 SAFSA 成功率为 82%。对目标函数 f_5 ~f_9,算法 LCAFSA 成功率均为 100%;对目标函数 f_5、f_7、f_8 和 f_9,算法 LCAFSA 迭代次数比算法 SAFSA 少很多;对目标函数 f_5、f_6、f_7 和 f_9,算法 LCAFSA 更加稳定。

以上结果说明,在指定精度下,算法 LCAFSA 的成功率、稳定性和计算效率均比算法 AFSA 和 SAFSA 有一定的优势。

3.3　其他改进算法寻优结果对比

其他文献中大多用目标函数 f_5 ~f_8 进行对比分析,这里针对目标函数 f_5 ~f_8 与其他改进算法寻优结果进行对比。表 5 为在固定迭代次数下算法 LCAFSA 优化结果与其他优化算法寻优结果对比。其中,固定迭代次数为 1 000 次。表 5 中"–"代表文献中未提到该目标函数优化结果。算法 AFSA 为基本人工鱼群算法;算法 SAFSA 为根据人工鱼觅食结果、种群中心位置和种群最优位置进化的一种简化的人工鱼群算法;算法 MAFSA 为动态调整视野及步长的改进人工鱼群算法;算法 MAAFSA 为多智能体人工鱼群算法;算法 IAFSA 为改进觅食算子和调整视野及步长的改进人工鱼群算法;算法 AFSABPCO 为基于比例选择算子的人工鱼群算法;算法 GAFSA 为全局版人工鱼群算法。以上算法及优化结果均来源于文献[20]。

表 5　固定迭代次数下 LCAFSA 优化结果与其他优化算法寻优结果对比

算法	函数			
	f_5	f_6	f_7	f_8
AFSA	0.549 31	0.720 78	31 291.7	137.326
SAFSA	1.15E – 08	0.118 24	4.098 545	8.183 427
MAFSA	–	0.062 2	0.113 494	0.045 879
MAAFSA	0	0.001 5	0.219 393	3.206 213
IAFSA	7.94E – 09	0.003 6	4.82E – 08	3.02E – 05
AFSABPCO	6.06E – 10	0.001 8	3.63E – 08	7.89E – 08
GAFSA	0	0.004 9	0.000 185	0.000 018
LCAFSA	0	1.54E – 02	2.55E – 09	0

从表 5 对比结果中可以看出,对目标函数 f_5,LCAFSA、MAAFSA 和算法 GAFSA 找到理论最优解,其余算法未找到理论最优解;对目标函数 f_6,算法 LCAFSA 优化结果精度处于均等水平;对目标函数 f_7,算法 LCAFSA 优化结果高于其余算法优化结果;对目标函数 f_8,算法 LCAFSA 可以得到理论最优解,其余优化算法均未得到。因此,算法 LCAFSA 与其他优化算法相比在求解精度方面有一定的优势。

3.4　调用目标函数次数和运行时间对比

表 6 为算法 AFSA 与 LCAFSA 对目标函数 f_1 ~f_9 运行到指定精度所需调用目标函数次数、计算梯度次数和运行时间的对比。目标函数指定精度同上,最大运行次数为 1 000 次。从表 6 可以看出,对目标函数 f_1,算法 LCAFSA 运算量约为算法 AFSA 的三分之一;对目标函数 f_2 和 f_9,算法 LCAFSA 运算量约为算法 AFSA 的二十分之一;对目标函数 f_3、f_6 和 f_8,算法 LCAFSA 运算量约为算法 AFSA 的一半;对目标函数 f_4,算法 LCAFSA 运算量与算法 AFSA 相差不大;对于目标函数 f_5,算法 LCAFSA 调用函数次数明显减少,且计算梯度次数较少,运算量降低较多,优势明显;对目标函数 f_7,算法 LCAFSA 运算量约为算法 AFSA 的四分之一。对目标函数 f_1 ~f_3、f_5 ~f_9,算法 LCAFSA 比 AFSA 运行时间节省很多。

表 6　指定精度下调用函数次数、计算梯度次数和运行时间对比

函数	AFSA		LCAFSA		
	目标函数	t	目标函数	梯度次数	t
f_1	9.38E + 05	4.112 186	2.57E + 05	1.57E + 04	3.472 582
f_2	9.39E + 05	3.809 544	1.13E + 04	168	0.124 801
f_3	9.19E + 05	6.595 722	3.91E + 05	2.00E + 04	5.837 557
f_4	6.98E + 05	5.091 873	5.61E + 05	1.99E + 04	6.274 672
f_5	4.26E + 05	3.371 962	164	1	0.010 401
f_6	4.06E + 05	5.098 113	2.58E + 05	2 075	3.432 022
f_7	2.86E + 05	2.455 456	5.24E + 04	1 701	0.710 584
f_8	3.83E + 04	0.638 044	1.13E + 04	427	0.166 921
f_9	4.26E + 05	3.257 301	2.25E + 04	1 097	0.409 503

由上可知,算法 LCAFSA 在运算量和运行时间上较算法 AFSA 都有一定的优势。

3.5 10 ~30 维函数优化实验

Rosenbrock 函数全局最优点处于一个平滑、狭长的抛物形山谷内,为优化算法提供的信息有限,使算法很难辨别搜索方向,查找最优解也变得很困难。

表 7 为人工鱼个体数为 20 时,算法 LCAFSA 与 IAFSA[20] 和 FPSO[21] 求解 Rosenbrock 函数的结果比较;表 8 为人工鱼个体数为 10 时,算法 LCAFSA 与 IAFSA 对 Rosenbrock 函数优化结果的比较。由表 7 和 8 可以看出,在人工鱼个体数分别为 10、20 时,10、20 和 30 维下,算法 LCAFSA 的优化结果均比算法 FPSO 和 IAFSA 优化结果精度高且稳定。

表 7 人工鱼个体数为 20 时 Rosenbrock 函数平均最优值

维数	迭代次数	FPSO	IAFSA	LCAFSA
10	1 000	66.014	4.865 1	2.55E -09
20	1 500	108.29	17.991	3.60E -13
30	2 000	183.80	23.254	1.34E -16

表 8 人工鱼个体数为 10 时 Rosenbrock 函数平均最优值

维数	迭代次数	IAFSA	LCAFSA
10	1 000	5.396 0	2.53E -09
20	1 500	19.927	3.60E -13
30	2 000	26.770	1.36E -16

3.6 分析参数对算法影响

影响算法性能的主要参数有人工鱼个数、视野、步长、拥挤度因子和觅食算子最大尝试次数,每个参数有不同取值。算法本身具有一定随机性,参数相同优化结果也会有一定差异。这里主要选取人工鱼个数、视野和步长分析其对算法结果的影响。

表 9 为当人工鱼个体数分别为 5、10、15、20、30 和 50 条,不同维度固定迭代次数时,算法 LCAFSA 对目标函数 f_5 ~f_9 的优化结果。当维数为 10 时,迭代次数为 1 000;当维数为 20 时,迭代次数为 1 500;当维数为 30 时,迭代次数为 2 000。表 10 为视野分别为 5、10、25、30、40、50、65、75、80、90,其他参数不变,算法 LCAFSA 对目标函数 f_5、f_7、f_9 的优化结果;表 11 为步长为 1、3、5、8、12、15、18、20、25,其他参数不变,算法 LCAFSA 对目标函数 f_5、f_7、f_9 的优化结果。

表 9 不同人工鱼个体数在固定迭代次数时算法 LCAFSA 优化结果

函数	维数	次数	$N=5$	$N=10$	$N=15$	$N=20$	$N=30$	$N=50$
f_5	10	1 000	0	0	0	0	0	0
	20	1 500	0	0	0	0	0	0
	30	2 000	0	0	0	0	0	0
f_6	10	1 000	0.361 055 243	3.44E -02	4.29E -02	1.54E -02	1.84E -02	1.18E -02
	20	1 500	0.241 512 248	7.99E -02	8.11E -02	1.68E -02	6.84E -02	2.88E -02
	30	2 000	0.310 792 739	1.53E -02	1.83E -02	4.02E -02	1.41E -02	2.00E -02
f_7	10	1 000	2.50E -09	2.53E -09	2.50E -09	2.55E -09	2.52E -09	2.54E -09
	20	1 500	3.25E -13	3.60E -13	3.59E -13	3.60E -13	3.60E -13	3.60E -13
	30	2 000	1.36E -16	1.36E -16	1.21E -16	1.34E -16	1.36E -16	1.36E -16
f_8	10	1 000	0	0	0	0	0	0
	20	1 500	0	0	0	0	0	0
	30	2 000	0	0	0	0	0	0
f_9	10	1 000	2.58E -02	9.95E -14	9.95E -14	9.95E -14	9.95E -14	9.95E -14
	20	1 500	2.34E -01	9.79E -15	9.79E -15	9.79E -15	9.79E -15	9.79E -15
	30	2 000	1.19E +00	5.34E -14	5.34E -14	5.34E -14	5.34E -14	5.34E -14

表 10 视野不同在固定迭代次数时算法 LCAFSA 优化结果

visual	f_5	f_7	f_9	visual	f_5	f_7	f_9
5	0	4.07E -07	1.07E -11	50	0	2.61E -07	1.51E -11
10	0	4.08E -07	5.45E -12	65	0	4.13E -07	1.11E -12
25	0	2.55E -09	9.95E -14	75	0	4.03E -07	3.25E -12
30	0	9.85E -08	1.16E -13	80	0	6.15E -07	1.57E -11
40	0	4.15E -08	9.85E -13	90	0	2.32E -07	2.08E -12

表 11 步长不同在固定迭代次数时算法 LCAFSA 优化结果

step	f_5	f_7	f_9	step	f_5	f_7	f_9
1	0	7.80E -08	1.58E -07	15	0	1.822 858	0.111 210
3	0	2.55E -09	9.95E -14	18	0	3.641 475	0.294 471
5	0	6.48E -08	1.83E -12	20	0	1.815 699	0.123 968
8	0	1.328 860	1.46E -12	25	0	3.149 728	4.59E -04
12	0	3.638 817	0.510 039				

由表 9 可看出,当 $N \geqslant 10$ 时,算法 LCAFSA 较稳定,优化结果相差不大,收敛精度都比较高;当 $N=5$ 时,优化目标函数 f_6、f_9 不稳定。人工鱼的数目越多,鱼群的群体智能越突出,收敛的速度越快,精度越高,跳出局部极值的能力越强;但是算法每次迭代的计算量也越大,计算效率越低。因此,在满足稳定收敛的前提下,应尽可能减少人工鱼的个体数量。由表 9 可看出,利用算法 LCAFSA 求解一般的函数优化问题时,人工鱼个体数在 10 ~20 条间比较合适。

由表 10 可以看出,对目标函数 f_5,visual 的调整对优化结果影响不大;对目标函数 f_7 和 f_9,visual 取 10 ~40 时优化结果相对较好,visual 取 25 时优化结果最好。视野范围较小时,觅食算子作用突出,聚群算子和追尾算子效果受到抑制;随着视野的不断增大,聚群算子和追尾算子发挥作用越来越大,觅食算子作用减小,人工鱼个体之间信息的相互作用逐步加强,有利于鱼群收敛于全局极值。

由表 11 可以看出,对目标函数 f_5,step 的调整对优化结果影响不大;对目标函数 f_7 和 f_9,step 取 1 ~5 时优化结果相对较好,当 step 为 3 时优化结果最好。当步长选择较小时,容易降低人工鱼收敛速度;随着步长逐步增大,收敛速度有一定的提升,精度一定程度降低;但是当步长达到一定程度后,会出现振荡现象,影响算法收敛精度。

综上所述,针对目标函数 f_5、f_7、f_9 选取参数取值为 $N=20$,visual = 25,step = 3,其他几个目标函数参数选取也依照此原则进行选取。

4 结束语

最速下降法是搜索函数最小值的经典优化算法,由于采用梯度信息,可以较快地收敛到函数极小值;基本人工鱼群算法在搜索区间广泛随机采点,然后从各点出发,依据鱼群觅食原理搜寻全局最小点,因而具有较好的全局搜索能力,但是此算法具有较大的随机性。本文引入最速下降法提出对精英加速的改进人工鱼群算法(LCAFSA)正是结合了两者的优点,利用最速下降法对适应度值最好的人工鱼进行更新,提高人工鱼整体水平,加强对其他人工鱼指导。算法 LCAFSA 无论从收敛精度、进化速度还是稳定性方面相比于其他人工鱼改进算法都有较好的优越性,但是对于非线性多模态函数优化效率还不是很理想,该问题还有待进一步研究并解决。

参考文献:

[1] 李晓磊. 一种新型的智能优化算法——人工鱼群算法[D]. 杭州: 浙江大学, 2003.

[2] 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002, 22(11): 32-38.

[3] 江铭炎, 袁东风. 人工鱼群算法及其应用[M]. 北京: 科学出版社, 2012: 20.

[4] 于飞, 张秋亮, 王智慧. 基于人工鱼群算法的研究与改进[J]. 中国电力教育, 2007(s3): 234-237.

[5] 王培崇, 李丽荣, 高文超, 等. 应用佳点集的混合反向学习人工鱼群算法[J]. 计算机应用研究, 2015, 32(7): 1992-1995.

[6] 吉鹏飞, 齐建东, 朱文飞. 改进人工鱼群算法在 Hadoop 作业调度算法的应用[J]. 计算机应用研究, 2014, 31(12): 3572-3574,3579.

[7] 任克强, 高晓林, 谢斌. 基于 AFSA 和 PSO 融合优化的 AdaBoost 人脸检测算法[J]. 小型微型计算机系统, 2016, 37(4): 861-865.

[8] 刘文礼, 陶佰睿, 张景林, 等. 基于广义高斯随机混沌算法的 AFSA 在 WSN 覆盖中的研究[J]. 计算机应用研究, 2015, 32(5): 1475-1479.

[9] 袁野, 杨红雨, 羽翼, 等. 人工鱼群—粒子群混合算法优化进港航班排序[J]. 计算机应用研究, 2014, 31(3): 663-666.

[10] 王明鸣, 孟相如, 李纪真, 等. 基于着色树优化的网络并发链路故障快速恢复方法[J]. 计算机应用研究, 2015, 32(6): 1822-1825.

[11] 高雷阜, 高晶, 赵世杰. 人工鱼群算法优化 SVR 的预测模型[J]. 统计与决策, 2015(7): 13-16.

(下转第 1981 页)

误率减少 78.7%。由实验结果可以看出,本文提出的基于特征层面校正类标的迁移学习算法有效地提高了分类的准确率。

本文提出的算法主要包括两个参数,一个是最大迭代次数 K,另一个是迭代终止参数 ε。二分类中各数据集迭代次数如图 2 所示。从图 2 可以看出,在 K 取值适度的条件下,算法对 K 值并不敏感,每一组实验中,该算法在迭代 5 ~ 16 次间都得到收敛。所以,本文根据经验值将 K 设置为 20。而迭代终止条件参数 ε 用于算法 2 中校正模糊隶属度。理论上,ε 越小,得到的结果越准确。所以,根据经验本文选取 0.05 作为 ε 的值。

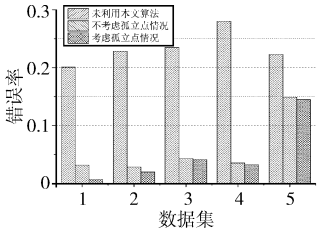

图 1　二分类数据集上的错误率

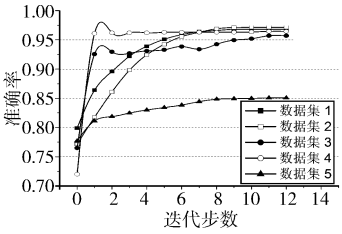

图 2　二分类中各数据集迭代次数

如图 3,4 所示,利用本文方法,在四分类数据集(表 2)上,分别对朴素贝叶斯和支持向量机的效果作了对比。对于朴素贝叶斯分类方法来说,未运用本文方法进行分类的正确率为 76.7%,而运用本文方法进行分类时,在第 3 次迭代后,结果达到稳定,分类的正确率为 83.1%,考虑孤立点后,分类的正确率为 85.5%;而对于支持向量机分类方法来说,未运用本文方法进行分类的正确率为 71.0%,而运用本文方法进行分类时,在第 6 次迭代后,结果达到稳定,分类的正确率为 94.8%,考虑孤立点后,分类的正确率为 95.3%。由结果看出,在处理该数据集时,不同的分类器会对本文方法产生影响,支持向量机效果明显好于朴素贝叶斯分类器。

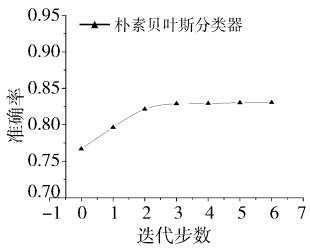

图 3　朴素贝叶斯作为分类器的四分类效果

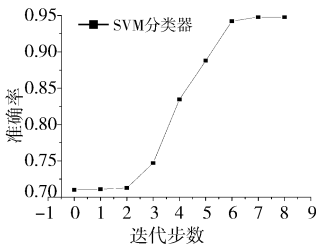

图 4　支持向量机作为分类器的四分类效果

4　结束语

本文提出了一种在训练集和测试集具有不同分布条件下通过迁移学习提高分类效果的算法。该方法利用最近自然邻居算法来计算测试集中样本的相似性,进而引用 FCM 算法对测试集类标进行校正。为了验证该算法的有效性,本文设计了两类实验,分别对二分类问题和四分类问题进行实验。结果表明,在处理二分类问题和多分类问题上,分类效果均有明显提高;同时,在四分类问题中,本文通过实验对比了朴素贝叶斯算法和支持向量机算法作为普通分类器对该算法的影响,结果表明,在处理四分类问题中,支持向量机表现的效果要优于朴素贝叶斯分类器。

在未来工作中,希望能够将该算法应用于网络大规模文本分类中,通过并行处理降低处理时间。同时,在迁移学习过程中,不仅仅考虑文本之间的相关性,而是将数据集扩展到图像数据与文本数据之间的联系,达到图像与文本之间迁移学习的目的。另外,在特征提取过程中,可以对比不同的方法对算法的影响。

参考文献:

[1] Huang Jiayuan, Smola A J, Gretton A, *et al.* Correcting sample selection bias by unlabeled data[C]//Proc of the 19th International Conference on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2007: 601-608.

[2] Ahmed A, Yu Kai, Xu Wei, *et al.* Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks [C]//Proc of the 10th European Conference on Computer Vision. Berlin: Springer, 2008: 69-82.

[3] Hubel D H, Wiesel T N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex [J]. Journal of Physiology, 1962, 160(1): 106-154.

[4] Dai Wenyuan, Xue Guiyong, Yang Qiang, *et al.* Transferring naive Bayes classifiers for text classification [C]//Proc of the 22nd National Conference on Artificial Intelligence. 2007:540-545.

[5] 张莹. 基于自然最近邻居的分类算法研究[D]. 重庆: 重庆大学, 2015.

[6] Yu Jianfei, Jiang Jing. A hassle-free unsupervised domain adaptation method using instance similarity features [C]//Proc of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. 2015: 168-173.

[7] Xing Dikan, Dai Weyuan, Xue Guiyong, *et al.* Bridged refinement for transfer learning [C]//Proc of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. Berlin: Springer-Verlag, 2007:324-335.

[8] Dunn J C. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters [J]. Journal of Cybernetics, 1973, 3(3): 32-57.

[9] Bezdek J C. Pattern recognition with fuzzy objective function algoritms [M]. Norwell, MA: Kluwer Academic Publishers, 1981.

[10] Peng Hanchuan, Long Fuhui, Ding C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2005, 27(8): 1226-1238.

[11] Miller G A, Beckwith R, Fellbaum C D, *et al.* WordNet: an online lexical database [J]. International Journal of Lexicography, 1990,3(4): 235-244.

(上接第 1964 页)

[12] Zhang Jiang, Nie Guojiang, Li Shouju. Study on mathematical model of confined transportation of coal auger and simulation[J]. Meitan Xuebao/Journal of the China Coal Society,2013,38(S1):231-235.

[13] 马骊, 李阳, 樊锁海. 改进人工鱼群算法在外汇预测和投资组合中的应用[J]. 系统工程理论与实践, 2015, 35(5): 1256-1266.

[14] 龚波, 曾飞艳. 一种改进人工鱼群算法对 BP 神经网络的优化研究[J]. 湖南科技大学学报: 自然科学版, 2016, 31(1): 86-90.

[15] 郭海湘, 刘嫣然, 杨娟, 等. 煤矿物资配送车辆路径问题的人工鱼群算法[J]. 系统管理学报, 2012, 21(3): 341-351.

[16] Li Xiaohua. Parameter optimization of lowest secondary crushing rate for coal auger based on artificial fish school algorithm[J]. Meitan Xuebao/Journal of the China Coal Society,2011,36(2):346-350.

[17] Jiang Mingyan, Wang Yong, Pfletschinger S, *et al.* Optimal multiuser detection with artificial fish swarm algorithm [C]//Proc of the 3rd International Conference on Intelligent Computing. Berlin: Springer, 2007: 1084-1093.

[18] 杨淑莹, 张桦. 群体智能与仿生计算——MATLAB 技术实现[M]. 北京: 电子工业出版社, 2014: 208.

[19] 马昌凤. 最优化方法及其 MATLAB 程序设计[M]. 北京: 科学出版社, 2010: 42.

[20] 王联国, 施秋红. 人工鱼群算法[M]. 北京: 中国农业出版社, 2014: 22.

[21] Shi Yuhui, Eberhart R C. Fuzzy adaptive particle swarm optimization [C]//Proc of Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press,2001: 101-106.