

自适应步长萤火虫优化算法

欧阳喆,周永权

(广西民族大学 数学与计算机科学学院, 南宁 530006)

(yongquanzhou@126.com)

摘要:针对基本萤火虫算法优化多峰函数时求解精度不高和后期收敛较慢的问题,引入萤光因子以自适应调整萤火虫的步长,提出一种自适应步长萤火虫优化算法。通过8个标准测试函数测试,测试结果表明,改进后的自适应步长萤火虫算法比基本萤火虫算法具有较快的寻优速度和较高的寻优精度。

关键词:多峰函数;萤火虫优化算法;自适应;步长;萤光因子

中图分类号:TP183 **文献标志码:**A

Self-adaptive step glowworm swarm optimization algorithm

OUYANG Zhe, ZHOU Yong-quan

(College of Mathematics and Computer Science, Guangxi University for Nationalities, Nanning Guangxi 530006, China)

Abstract: According to the problem that Glowworm Swarm Optimization (GSO) cannot acquire solutions exactly and converge slowly in the later period for solving the multimodal function, an improved GSO algorithm combined with luciferin-factor, which can adaptively adjust step, was proposed. The simulation results show that the improved Self-Adaptive Step Glowworm Swarm Optimization (ASGSO) can search for global optimization more quickly and precisely.

Key words: multimodal function; Glowworm Swarm Optimization (GSO) algorithm; self-adaptive; step; luciferin-factor

0 引言

萤火虫优化(Glowworm Swarm Optimization, GSO)算法是2005年由K. N. Krishnanand和Debasish Ghose提出的一种新的群智能优化算法。GSO主要是模拟萤火虫发光吸引同伴,萤火虫发光越大,吸引的同伴越多这一现象,通过各个萤火虫个体,在视野范围内寻找最亮的萤火虫,向最亮的萤火虫移动来实现寻优的目的^[1-2]。

多峰函数一般是指有多个峰值点的函数^[3]。多峰函数的优化问题在实践中大量存在,例如求解非线性方程组,依据密度的山峰聚类算法、神经网络训练等。但在实际中,有时无法求出全局最优值点作为最优方案,因此求出在局部最优值点函数作为次优解。因此需要进行多峰优化,找到函数全局最优值的同时也能找到多个局部最优值,但许多经典的算法往往容易陷入局部最优^[4]。所以研究有效快速的多模态函数的优化算法成为一个研究热点。

基本萤火虫算法在优化多峰函数时,能够同时获得多个极值点。但是算法后期收敛速度慢,寻优精度不高,降低了算法的效率。针对这些不足,本文提出了基于自适应步长的萤火虫算法(self-Adaptive Step Glowworm Swarm Optimization, ASGSO),引入了萤光因子对萤火虫算法的步长进行自适应调整。仿真结果表明,自适应步长的萤火虫算法具有计算精度较高、搜索速度较快等特点。

1 基本萤火虫算法

在基本GSO中,把 n 个萤火虫个体随机分布在一个 D 维目标搜索空间中。每个萤火虫都携带了萤光素 l_i 。萤火虫个体都发出一定量的萤光相互影响周围的萤火虫个体,并且拥

有各自的决策域 $r_d^i(0 < r_d^i \leq r_s)$ 。萤火虫个体的萤光素大小与自己所在位置的目标函数有关,萤光素越大,越亮的萤火虫表示它所在的位置越好,即有较好的目标值。萤火虫会在决策域内寻找邻居集合 N_i ,在集合中,萤光素越大的邻居拥有越高的吸引力,吸引萤火虫往这个方向移动,每一次移动的方向会随着选择的邻居不同而改变。另外,决策域的大小会受到邻居数量的影响,邻居密度越小,萤火虫的决策半径会加大以便寻找更多的邻居;邻居密度越大,它的决策半径则会缩小。最后,大部分萤火虫会聚集在多个位置上。初始萤火虫时,每个萤火虫个体都携带了相同的萤光素浓度 l_0 和感知半径 r_0 。

萤光素更新 每只萤火虫 i 在 t 迭代的位置 $x_i(t)$ 对应的目标函数值 $J(x_i(t))$ 转化为萤光素值 $l_i(t)$:

$$l_i(t) = (1 - \rho)l_i(t-1) + \gamma J(x_i(t)) \quad (1)$$

其中 γ 为萤光素更新率。

概率选择 每个个体在其动态决策域半径 $r_d^i(t)$ 内,选择萤光素值比自己高的个体组成其邻域集 $N_i(t) = \{j; d_{ij}(t) < r_d^i(t); l_i(t) < l_j(t)\}$,其中 $0 < r_d^i(t) \leq r_s$, r_s 为萤火虫个体的感知半径。选择移向邻域集 $N_i(t)$ 内个体 j 的概率 $p_{ij}(t)$:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} (l_k(t) - l_i(t))} \quad (2)$$

位置更新:

$$x_i(t+1) = x_i(t) + s \left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \right) \quad (3)$$

其中 s 为移动步长。

动态决策域半径更新:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_i - |N_i(t)|)\}\} \quad (4)$$

收稿日期:2011-01-10;修回日期:2011-03-02。 基金项目:广西自然科学基金资助项目(0991086)。

作者简介:欧阳喆(1988-),女,江西吉安人,硕士研究生,主要研究方向:智能计算;周永权(1962-),男,陕西旬邑人,教授,博士,主要研究方向:神经网络、计算智能。

简单地说,GSO 算法主要包括萤火虫的初始分布、萤光素更新、萤火虫的移动和决策域的更新四个阶段^[5]。

2 自适应步长萤火虫算法

2.1 动态自适应步长调整策略

在基本萤火虫算法中,存在容易陷入局部最优,后期收敛较慢且求解精度不高的问题。在搜索的过程中,步长越大,越容易搜索全局最优,但同时降低了搜索精度,有时会出现震荡现象;步长越小,搜索速度降低,求解精度提高。针对这个问题,需要根据不同阶段的搜索结果,动态调整步长的大小,处理好全局寻优能力和寻优精度之间的关系。

为此,引入萤光因子

$$H_i = \frac{\|X_i - X_{\text{ext}}\|}{d_{\text{max}}} \quad (5)$$

式中: X_i 表示第 i 只萤火虫个体的状态, X_{ext} 表示此时萤光素浓度最大的萤火虫个体的状态, d_{max} 表示最优萤火虫距其余所有萤火虫的距离的最大值。

基于萤光因子的自适应调整步长策略:

$$s_i = s_{\min} + (s_{\max} - s_{\min})H_i \quad (6)$$

式中 s_{\max} 和 s_{\min} 分别表示步长的最大值和最小值。

根据式(5)、(6)动态调整步长。当第 i 只萤火虫越接近萤光素浓度最大的萤火虫,距离越近,则 $\|X_i - X_{\text{ext}}\|$ 越小,根据式(5),所求得的 H_i 值越小,因此根据式(6)得出的 s_i 也越小。当萤火虫个体向目标萤火虫个体移动时,一个较小的步长 s_i 可以让萤火虫个体更好地靠近目标萤火虫个体,而不至于由于步长太大而跳过最优解,从而提高搜索精度。当第 i 只萤火虫个体距离萤光素浓度最大的萤火虫个体较远时,则 $\|X_i - X_{\text{ext}}\|$ 越大, H_i 值越大, s_i 值增大,当萤火虫个体以较大步长搜索时,可提高搜索速度,更快搜索寻优。这样,根据上一次迭代的结果来动态更新本次迭代的移动步长,具有更好的自适应性。同时,给定步长的最大值和最小值 s_{\max} 和 s_{\min} ,让步长在一定范围内变化,可以避免萤火虫分散过大导致步长太小等问题。对于标准的萤火虫算法,由于步长 s 固定,在初始化 s 时,随机性强,步长太大则会影响求解精度,太小则又降低搜索速度。所以,只有根据求解的情况动态地调整步长,才能更好地提高算法的搜索速度和寻优精度。

因此,在自适应步长的萤火虫算法运行前期,采用较大的步长,可以保证离最优萤火虫较远的个体具有更大的步长,使萤火虫在大范围内进行粗搜索,从而可以更快搜索全局最优的邻域;而在最优邻域附近的个体具有较小的步长,算法逐步演化为局部搜索进行精确搜索,可以更精确地搜索极值^[6-9]。

2.2 算法描述

基于自适应步长的萤火虫算法的流程可以描述如下。

步骤1 初始化群体:随机生成 n 个萤火虫个体形成初始萤火虫种群,搜索空间维数 m ,萤火虫的初始萤光素大小 l_0 和感知范围 r_0 ,萤火虫感知最大半径 r_s ,最大迭代次数 $iter_max$,初始步长 $s_i(0)$,最大步长 s_{\max} ,最小步长 s_{\min} ,随机产生萤火虫初始位置 $x_i(0)$,萤光素挥发系数 ρ ,萤光素更新率 γ ;

步骤2 对种群中每一个萤火虫个体 i ,执行以下操作:

2.1)使用式(1)把萤火虫 i 在第 t 次迭代的位置 $x_i(t)$ 对应的目标函数值 $J(x_i(t))$ 转化为萤光素值 $l_i(t)$;

2.2)每只萤火虫在其动态决策域半径 $r_d^i(t)$ 内,选择萤光素值比自己高的个体组成其邻域集 $N_i(t)$,其中 $0 <$

$r_d^i(t) \leq r_s$, r_s 为萤火虫个体的感知半径;

2.3)利用式(2)计算萤火虫 i 移向邻域集内个体 j 的概率 $p_{ij}(t)$;

2.4)利用轮盘赌的方法选择个体 j ;

2.5)据式(5)计算每个萤火虫个体的萤光因子,用式(6)计算移动步长;

2.6)进行移动,根据式(3)更新位置;

2.7)根据式(4)更新动态决策域半径的值。

步骤3 判断算法是否到达指定的最大迭代次数,如果是则转向步骤4,否则转向步骤2;

步骤4 输出结果,程序结束。

3 实验结果与分析

3.1 实验操作平台

本实验的程序运行环境为:处理器 CPU T5300,主频 1.73 GHz,内存 1 GB,操作系统:Windows XP,集成开发环境为 Matlab 7.0。

3.2 实验结果

算法参数设计如下:萤火虫算法种群规模 n 为 50,萤光素挥发系数 $\rho = 0.4$,萤光素更新率 $\gamma = 0.6$,初始萤光素大小 $l_0 = 5$,感知范围 $r_0 = 10$,初始步长 $s_i(0) = 0.03$,最大步长 $s_{\max} = 1$,最小步长 $s_{\min} = 10^{-4}$, $\beta = 0.08$, $n_t = 5$,最大迭代次数 200。粒子群算法种群规模 n 为 50,惯性权重 $\omega = 0.5$,学习因子 $c_1 = c_2 = 2$,最大迭代次数 200。

对函数 $F_1(x) \sim F_8(x)$ 进行 20 次独立实验^[10],维数为 10。表 1 列出了这些标准函数的名称及最优值等参数。这 8 个测试函数如下:

$$F_1(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)] + 10$$

$$F_2(x) = \sum_{i=1}^n (0.2x_i^2 + 0.1x_i^2 \sin 2x_i)$$

$$F_3(x) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

$$F_4(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$$

$$F_5(x) = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$$

$$F_6(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2)^{0.25} \times [\sin(50 \times (x_i^2 + x_{i+1}^2)^{0.1}) + 1]$$

$$F_7(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$$

$$F_8(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) \cos(4\pi x_2) + 0.3$$

表 1 标准测试函数

函数	函数名	搜索范围	最小值
$F_1(x)$	Rastigrin	$[-5.12, 5.12]^n$	0
$F_2(x)$	Function 15	$[-10, 10]^n$	0
$F_3(x)$	Griewank	$[-600, 600]^n$	0
$F_4(x)$	Rosenbrock	$[-50, 50]^n$	0
$F_5(x)$	Schaffer F6	$[-100, 100]^n$	0
$F_6(x)$	Schaffer's f7	$[-100, 100]^n$	0
$F_7(x)$	Bohachevsky 1	$[-50, 50]^n$	0
$F_8(x)$	Bohachevsky 2	$[-50, 50]^n$	0

表 2 列出算法优化函数的最差值、最优值、平均值和平均

迭代次数,并与基本萤火虫算法及粒子群算法(Partical Swarm Optimization, PSO)的结果进行比较,其中平均迭代次数为分别进行20次独立实验所求迭代次数的平均值。

表2 GSO与ASGSO,PSO的性能比较

测试函数	算法	最优值	最差值	平均值	平均迭代次数
$F_1(x)$	GSO	8.75E-06	4.65E-04	7.29E-05	121
	ASGSO	0	6.25E-06	4.14E-07	95
	PSO	2.98	9.94	7.76	200
$F_2(x)$	GSO	2.12E-12	7.14E-08	5.81E-09	115
	ASGSO	4.89E-32	3.46E-12	6.56E-13	100
	PSO	8.93E-10	4.09E-08	1.46E-09	152
$F_3(x)$	GSO	9.38E-10	1.32E-08	9.04E-08	107
	ASGSO	0	3.77E-14	6.340E-15	82
	PSO	9.75E-10	5.15E-06	8.28E-07	123
$F_4(x)$	GSO	2.29E-05	2.79E-03	6.44E-04	130
	ASGSO	5.84E-07	6.51E-04	1.60E-05	137
	PSO	5.04	9.47	6.82	200
$F_5(x)$	GSO	2.35E-06	1.15E-04	1.08E-05	122
	ASGSO	1.51E-08	3.16E-06	5.24E-07	101
	PSO	0	1.63E-12	6.15E-13	84
$F_6(x)$	GSO	7.83E-02	2.19E-01	1.45E-01	113
	ASGSO	5.30E-03	3.31E-02	1.59E-02	99
	PSO	7.29E-08	9.52E-07	9.01E-07	185
$F_7(x)$	GSO	4.68E-03	5.56E-02	2.37E-02	109
	ASGSO	1.93E-09	1.01E-05	2.05E-06	105
	PSO	0	4.44E-16	2.66E-15	70
$F_8(x)$	GSO	9.18E-03	1.68E-01	2.52E-02	137
	ASGSO	4.35E-09	9.68E-05	4.25E-05	127
	PSO	0	4.99E-10	5.55E-11	62

图1~8分别为GSO和ASGSO求函数 $F_1(x)$ ~ $F_8(x)$ 的最小值时函数随迭代次数变化。

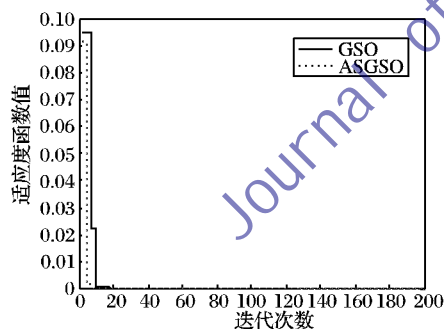


图1 函数 $F_1(x)$ 函数值随迭代次数变化

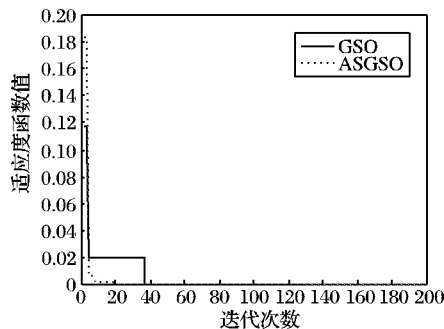


图2 函数 $F_2(x)$ 函数值随迭代次数变化

近标准值。对于粒子群算法,基本萤火虫算法和自适应步长的萤火虫算法在高维空间求解能力更强,粒子群算法在二维空间有比较好的搜索能力。对于高维尖峰函数 $F_1(x)$,ASGSO的最优值为0,求解精度比GSO提高了2个数量级,平均迭代次数减少了26次;而粒子群算法在最大迭代次数内,没有求出理想的解,搜索速度和搜索能力都不及GSO和ASGSO。对于函数 $F_2(x)$,ASGSO的最优值精度达到 e^{-32} ,平均求解精度比GSO提高了4个数量级,平均迭代次数减少了15次;而粒子群算法的最优值精度为 e^{-10} ,比GSO和ASGSO的最优精度都低。对于函数 $F_3(x)$,ASGSO的最优值为0,平均求解精度比GSO提高了7个数量级,平均迭代次数减少了一半;而粒子群算法的最优解的精度和基本萤火虫算法相同。对于函数 $F_4(x)$,ASGSO的最优值精度达到 e^{-7} ,平均求解精度比GSO提高了1个数量级,但是平均迭代次数相较GSO稍微多几代;粒子群算法在最大迭代次数内,没有求解出来。对于函数 $F_5(x)$,ASGSO的最优值精度达到 e^{-8} ,平均求解精度比GSO提高了2个数量级,平均迭代次数减少了20次;粒子群算法最优值为0,求解精度和速度都明显高于GSO和ASGSO。对于函数 $F_6(x)$,ASGSO的最优值精度达到 e^{-3} ,平均求解精度比GSO提高了1个数量级,平均迭代次数减少了14次;粒子群算法有较高的求解精度,比ASGSO高了5个数量级。二维多峰函数 $F_7(x)$ 和 $F_8(x)$,ASGSO的最优值求解精度都达到 e^{-9} ,平均精度都比GSO提高了4个和3个数量级,迭代次数也有所减少;粒子群算法在二维空间就更具有优势,最优值为0,求解精度提高,迭代次数减少。对于ASGSO和GSO,在高维空间求解多峰函数时,相比于粒子群算法具有更高的精度和速度,ASGSO的寻优能力比GSO更高些;而在二维空间时,粒子群算法的求解精度和寻优速度要比ASGSO和GSO要高很多。

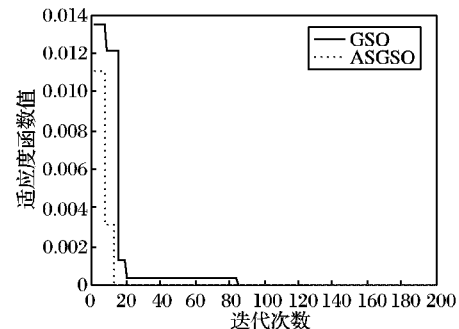


图3 函数 $F_3(x)$ 函数值随迭代次数变化

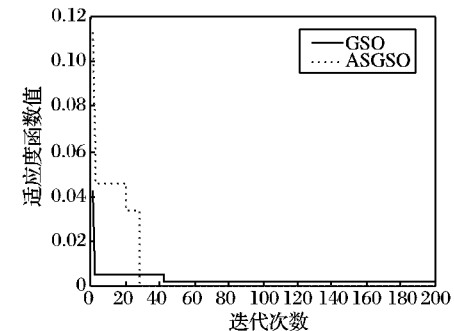


图4 函数 $F_4(x)$ 函数值随迭代次数变化

从图1可看到ASGSO收敛曲线的下降速度比GSO要快得多,且很快找到最优值,而GSO在第30~第175代很长一段时间陷入局部最优才找到最优值。从图2~图5可以看出,ASGSO的收敛速度明显高于GSO,GSO在一定的迭代次

3.3 结果分析

从表1中可以看出,与基本萤火虫算法相比,自适应步长的萤火虫算法搜索最优值的能力更好,求解的最优值也更接

数中陷入局部最优很难跳出,而 ASGSO 由于动态调整了步长能够更快地跳出局部最优更快收敛。从图 6 可以看出,ASGSO 求解精度比 GSO 提高了很多。图 7 和图 8,ASGSO 的收敛曲线下降得比 GSO 要快,能很快地收敛到全局最优。动态调整步长,可以使算法在收敛初期更快寻优,在陷入局部最优时能很快跳出局部最优,使 ASGSO 能比 GSO 更快地收敛到全局最优并提高解的精度。

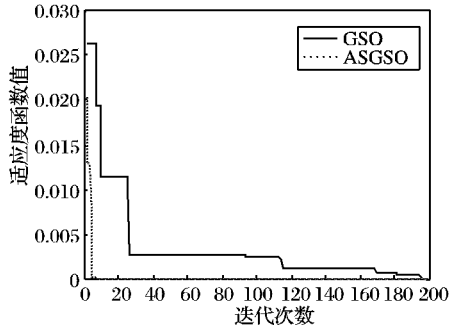


图5 函数 $F_5(x)$ 函数值随迭代次数变化

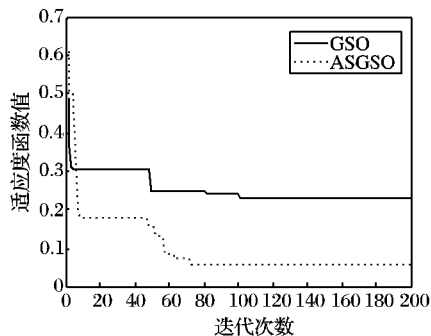


图6 函数 $F_6(x)$ 函数值随迭代次数变化

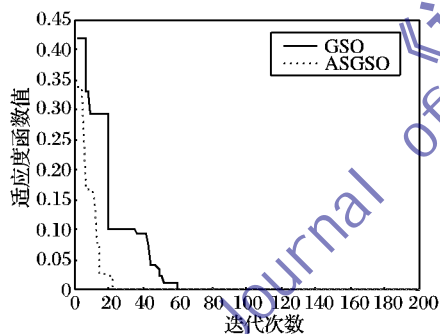


图7 函数 $F_7(x)$ 函数值随迭代次数变化

过萤光因子来动态调整步长,使算法避免陷入局部最优,提高寻优速度和精度。通过函数优化实验可以看出,算法是可行的,而且优于基本的萤火虫算法。不仅进行迭代的次数减少了,提高了进化速度,而且获得了更精确的函数值。关于算法的收敛性证明,算法参数的优化及算法的其他应用需要深入研究,也是下一步将要要做的工作。

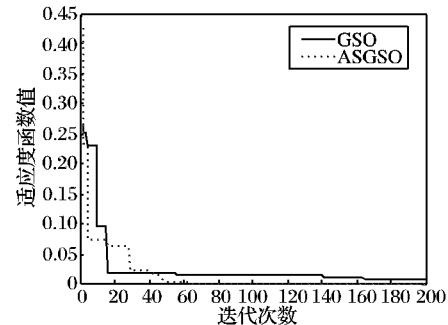


图8 函数 $F_8(x)$ 函数值随迭代次数变化

参考文献:

- [1] KRISHNAND K N, GHOSE D. Glowworm swarm optimisation for simultaneous capture of multiple local optima of multimodal functions [J]. Swarm Intelligence, 2009, 3(2): 87-124.
- [2] KRISHNAND K N, GHOSE D. Glowworm swarm optimisation: A new method for optimising multi-modal functions [J]. The International Journal of Computational Intelligence Studies, 2009, 1(1): 93-119.
- [3] CASTROL, ZUBEN L. Learning and optimization the clonal selection principle evolution computation [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(3): 34-50.
- [4] 赵吉, 孙俊, 须文波. 一种求解多峰函数优化问题的量子行为粒子群算法 [J]. 计算机应用, 2006, 26(12): 2956-2960.
- [5] 曾国泰. 萤火虫最佳化分群演算法 [D]. 台北: 大同大学资讯经营所, 2008.
- [6] 刘彦君, 江铭炎. 自适应视野和步长的改进人工鱼群算法 [J]. 计算机工程与应用, 2009, 45(25): 35-37.
- [7] 王联国, 洪敏, 赵付青, 等. 一种改进的人工鱼群算法 [J]. 计算机工程, 2008, 34(19): 192-194.
- [8] 王航, 薛晓中, 孙瑞胜. 基于人工鱼群算法的制导炸弹控制系统设计与仿真 [J]. 弹道学报, 2009, 28(4): 98-102.
- [9] 王晔, 吴小俊, 王士同. 基于改进人工鱼群算法的 RBF 网络及其在人脸表情识别中的应用 [J]. 计算机应用研究, 2008, 25(9): 2643-2646.
- [10] DEEP K, BANSAL J C. Mean particle swarm optimization for function optimization [J]. The International Journal of Computational Intelligence Studies, 2009, 1(1): 72-92.

4 结语

针对基本 GSO 优化多峰函数时早熟停滞的问题,提出了自适应步长的萤火虫算法。算法中引入了萤光因子概念,通

(上接第 1796 页)

的泛化能力,能更加有效地预测地震,使地震灾害损失降到最低。

参考文献:

- [1] 王伟, 蒋春曦. BP 神经网络在地震综合预报中的应用 [J]. 地震, 1999, 19(2): 118-126.
- [2] 秦长源, 张淑亮. AR 模型预测太原盆地地震活动的趋势 [J]. 地震研究, 1997, 20(4): 4-9.
- [3] 陈一超, 曾三友, 张好春等. 基于遗传神经网络的地震预测研究 [J]. 计算机应用与软件, 2008, 25(4): 77-85.
- [4] 武安绪, 李平安, 鲁亚军, 等. 基于支持向量机的多维地震时间序列建模 [J]. 东北地震研究, 2006(4): 4.

- [5] 杨居义, 易永红. 基于 BP 神经网络的地震预测研究 [J]. 微电子学与计算机, 2008, 25(10): 11-15.
- [6] 高海兵, 高亮, 周驰等. 基于粒子群优化的神经网络训练算法研究 [J]. 电子学报, 2004, 32(9): 15-19.
- [7] 张立, 平建军, 苏有锦. 云南地区综合地震前兆信息量及其短期映震能力分析 [J]. 地震研究, 2006, 29(10): 26-29.
- [8] 王凤, 黄力宇, 张宇翔. 基于 Matlab 的 BP 预测模型在地震前兆预测中的应用研究 [J]. 华北地震科学, 2009, 27(1): 48-52.
- [9] 王存睿, 段晓东. 改进的基本粒子群优化算法 [J]. 计算机工程, 2004, 30(21): 55-63.
- [10] 王晓霞, 王涛, 谷根代, 等. 基于改进粒子群优化的神经网络及应用 [J]. 华北电力大学学报, 2009, 36(5): 99-102.