

一种改进的萤火虫算法

左仲亮¹, 郭 星², 李 炜²

(1 安徽大学 计算智能与信号处理重点实验室, 安徽 合肥 230039;

2 安徽大学 计算机科学与技术学院, 安徽 合肥 230601)

摘 要: 为了克服原始萤火虫算法(Glowworm swarm optimization, GSO)对于高维、多峰函数寻优精度不高和后期收敛速度较慢的问题, 为此, 有针对性地提出了一种改进的动态步长萤火虫优化算法, 在整个迭代期间, 萤火虫的算法步长呈非线性递减, 在寻优初期保持着一个相对较大的步长进行搜索, 增强其全局寻优能力, 在算法后期保持一个较小的移动步长, 增强其局部搜索能力。此外将原始萤火虫算法在 Nit 集合为 0 时不移动, 改成试探性向外随机移动策略。采用该算法的改进思想, 能在一定的程度上避免算法因为过早的成熟而陷入局部最优, 并且改进的算法比原始萤火虫算法有着更好的收敛精度。通过与原始 GSO 和改进算法 ASGSO 做对比, Matlab 实验仿真表明, 此改进算法在一定程度上提高了收敛速度和寻优精度。

关键词: 萤火虫算法; 多峰函数; 动态步长

中图分类号: TP391.9

文献标识码: A

文章编号: 1000-7180(2018)02-0061-06

An Improved Swarm Optimization Alogorithm

ZUO Zhong-liang¹, GUO Xing², LI Wei²

(1 Key Laboratory of ICSP, Ministry of Education, Anhui University, Hefei 230039, China;

2 School of Computer Science and Technology, Anhui University, Hefei 230601, China)

Abstract: In order to overcome the basic artifical firefly algorithm(GSO) in solving problems of low precision for the multi peak function, high dimension and slow convergence. Thus, this paper comes up with a Dynamic step optimization algorithm. During the iteration, the step decrease nonlinearly. At the beginning of the optimization maintains a large step to search optimal value, enhancing global optimization ability. At the ending of it, optimization maintains a small step to improve ability of local searching. In addition, original firefly algorithm does not move anywhere, when Nit is null, this paper makes it move somewhere randomly. With it, This algorithm can avoid being mature and falling into a local value. As the same time, improved GSO can achieve a higher accuracy during Iterative process, comparing with GSO and reference literature alogorithm ASGSO. According to the simulation experiment, it shows that to some extent, the improved algorithm on covergence speed and precision are enhanced.

Key words: GSO; multi peak function; dynamic step

1 引言

人工萤火虫算法(Glowworm Swarm Optimization)GSO 算法, 主要是通过模拟萤火虫发光来吸引其他萤火虫同伴, 荧光素浓度越大, 萤火虫亮度越大, 吸引同伴就越多。初始萤火虫种群分布在随机的位置, 萤火虫依据自己的感知范围, 寻找具有最亮的萤火虫, 并向最亮的萤火虫移动。通过不断的迭

代计算, 来达到寻优的目的^[1]。目前, 萤火虫算法被成功应用到函数的最优解上面。此外, 萤火虫算法在其他领域也有诸多用途, 该算法具有操作简单, 鲁棒性强, 易于实现等优点。但也存在容易早熟, 陷入局部极值点和震荡的缺点。针对这些缺点, 本文提出一种动态步长萤火虫算法, 仿真实验表明, 改进后的萤火虫算法比原始算法有着更快的收敛速度, 和更高的收敛精度。

2 基本萤火虫算法介绍

2.1 算法描述

在原始 GSO 算法中,会把 N 只萤火虫个体随机地分布在一个 D 维的搜索空间. 每一只萤火虫都会携带荧光素 l_i , 荧光素的值与萤火虫的亮度成正比关系. 萤火虫个体的荧光素大小由个体所处的位置的目标函数决定, 荧光素越大越亮的萤火虫表示该个体所处的位置较好, 有着较优的目标函数值. 每只萤火虫会在自己的决策域 r_d^i (其范围是 $[0, rs]$) 去寻找邻居萤火虫集合 N_i 在该集合中, 荧光素越大的邻居拥有越强的吸引力, 吸引其他萤火虫朝着自己的方向移动. 每一次移动方向都会随着领域内最亮萤火虫不同而发生改变. 此外, 萤火虫决策阈 r_d^i 随着邻居的数量而改变. N_i 值越小, 萤火虫的决策阈适当增大, 去寻找更多的萤火虫邻居, 反之则会缩小决策阈. 最后通过算法的不断迭代, 最优值的位置会聚集着较多的萤火虫, 从而达到寻优的目的.

2.2 原始 GSO 算法主要步骤

GSO 算法主要分为 5 个步骤: 萤火虫个体初始化, 荧光素更新, 轮盘赌概率选择, 位置更新, 决策阈更新.

(1) 萤火虫个体初始化

对算法的萤火虫个体各个参数例如荧光素挥发因子, 适应度提取比例, 邻域变化率, 邻域阈值, 移动步长, 荧光素浓度, 感知半径, 决策半径, 进行初始化.

(2) 荧光素更新

萤火虫的荧光素值与其在目标函数的位置有关, 目标函数值越优, 表明该个体的适应度就越好, 则计算后个体的荧光素值就越大. 由于萤火虫的亮度是随着荧光素的大小来决定的, 所以它的亮度也就越强. 根据计算公式可知, 萤火虫的亮度还与发散因子大小和上一次迭代得到的荧光亮度有关. 其更新公式如下:

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

式中, ρ 的取值范围是 $(0, 1)$, 代表萤火虫的荧光素在空气中的挥发因子; $l_i(t)$ 代表第 i 只萤火虫在第 t 次迭代荧光素大小; γ 是萤火虫适应度提取比例; $J_i(x_i(t+1))$ 是代表萤火虫 i 在 $t+1$ 次迭代的位置 $x_i(t+1)$ 对应的目标函数值.

(3) 轮盘赌概率选择

萤火虫算法在每一次迭代之后, 每只萤火虫都会在其动态决策域半径 r_d^i 内, 选择一个萤火虫并朝

其移动, 在进行移动之前, 选择荧光素值比自己高的萤火虫个体组成一个邻居集合 $N_i(t)$. 成为萤火虫 i 的邻居需要同时满足以下两个条件: 第一, 萤火虫 j 必须在萤火虫的 i 的决策域范围内; 第二, 在此范围内的萤火虫 j 的荧光素值必须比萤火虫 i 大. 在邻居集合 $N_i(t)$ 中, 利用轮盘赌法则选择一个合适的萤火虫, 让萤火虫 i , 朝着其移动. 邻居集合内每个萤火虫被选择的概率计算如下:

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

式中, $j \in N_i(t)$, $N_i(t) = \{j: d_{i,j}(t) < r_d^i; l_j(t) < l_i(t)\}$, $d_{i,j}(t)$ 代表在第 t 次迭代萤火虫 i 与萤火虫的 j 的空间距离. 依据轮盘赌法则在邻居集合中选出一只萤火虫的 j .

(4) 萤火虫位置更新

当 i 的邻居萤火虫被选出之后就可以进行第 $t+1$ 次位置的更新. 依据式(3) 进行更新:

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

式中, s 是萤火虫的固定移动步长; $||x_j(t) - x_i(t)||$ 代表萤火虫 j 与萤火虫 i 的欧式距离.

(5) 萤火虫动态决策域更新

在萤火虫位置更新后, 会动态地去更新自身的决策阈大小. 决策域内邻居数量的大小决定了更新的大小. 依据式(4) 进行更新:

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

式中, β 是调整萤火虫动态决策域的调节因子, 是一个设定好的参数; n_i 是领域集合大小的阈值, 也是预先设定好的.

3 原始萤火虫算法缺陷及改进算法

缺陷一: 在最基本的 GSO 算法中, 萤火虫移动是采用一个固定步长大小进行的. 在对目标区间搜索的过程中, 若步长取得比较大, 则越容易搜索全局最优值, 但同时降低了最优值的精度, 而且容易在最优值附近点剧烈震荡. 若步长保持得较小, 能够得到一个比较优的精度目标值, 但是搜索速度太慢, 需要很多次的迭代, 有时候因为移动能力太弱的原因, 无法找到最优值.

缺陷二: 在原始萤火虫算法当中, 萤火虫 i 的移动是依靠着在其决策域范围内找到比自身荧光素大的萤火虫 j . 这些萤火虫组成一个邻居集合, 萤火虫

i 会依据轮盘赌法在集合选择中一只萤火虫,以固定步长朝向被选择出的萤火虫移动. 原始萤火虫算法对于邻居集合为空时采取的是不移动. 邻居集合为空意味着萤火虫 i 在其决策域范围内没有发现比自己荧光素更大的萤火虫. 这会引发一个问题:当这只萤火虫处在离最优值比较远的地方时,它周围又没有更优的萤火虫,它会保持不移动,周围萤火虫看见它比较亮,朝其移动,最后造成一部分萤火虫集中在一个位置,并且丧失了移动能力. 这种情况可用图 1 表示:大圆代表萤火虫 i ,在它的周围是比自己荧光素值低的萤火虫. 三角形代表最优值的位置. 萤火虫 i 找不到比自己亮的,会保持不动. 周围的萤火虫将会聚集在 i 的旁边,导致一部分萤火虫丧失移动能力,算法寻优效率下降.

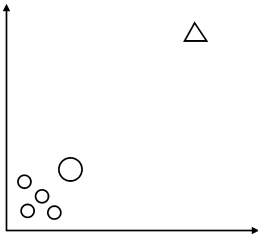


图 1 Nit 集合为空的情况

3.1 基于动态步长的改进萤火虫优化算法

根据缺陷 1,本文采用式(5)来动态地改变萤火虫的移动步长. 在萤火虫算法迭代初期,萤火虫保持着一个较大的步长,保持着一个较好的全局寻优能力,避免萤火虫过早的成熟,陷入局部极值. 在萤火虫算法迭代后期,萤火虫的移动步长逐渐变小,使其在局部范围内精细搜索,有利于萤火虫算法在后期的收敛,不会因为移动步长过大导致跳过最优解. 根据缺陷 2. 当萤火虫的邻居集合为空时,代表它自身荧光素相对于决策域内的萤火虫较多,但不能代表这只萤火虫处在一个离最优极值比较近的位置. 所以当邻居集合为空时,萤火虫 i 以随机步长的方式尝试运动 20 次,在这 20 次当中选择一个最优的位置去更新萤火虫 i 的位置. 这样能够以很大的概率避免一批萤火虫丧失移动能力的情况出现.

$$s(t) = s_{\max} * e^{-p * q^{1.5}} + s_{\min} \quad (5)$$

$$q = \frac{t}{t_{\max}} \quad (6)$$

$$x_i(t+1) = x_i(t) + rand(1, D) * rand(1) \quad (7)$$

在上述 3 个公式当中, t 代表迭代次数; $t_{\max}$ 代表最大迭代次数; $s_{\max}$, $s_{\min}$ 分别代表最大移动步长和最小移动步长; P 是步长变化幅度调节因子; D 代表萤

火虫个体的维度.

3.2 改进 GSO 算法的步骤

为方便起见,本文改进后的算法称为 DSGSO.

步骤一:初始化萤火虫种群数量 n ,并将萤火虫随机分布在问题的搜索空间中. 初始萤火虫最大移动步长 $s_{\max}$ 和最小移动步长 $s_{\min}$,迭代次数 $iteration$,和 $\rho, \gamma, \beta, \eta, s, \epsilon_0, r_0, \rho$ 等参数.

步骤二:萤火虫的位置更新之后,在算法下一次迭代前,整个萤火虫种群的荧光素都会得到更新,使用式(1) 将萤火虫 i 在第 t 次迭代生成的位置 $x_i(t)$ 所对应的适应度值 $J(x_i(t+1))$,换算成萤火虫的荧光素值 $I_i(t+1)$.

步骤三:算法进入迭代寻优阶段.

步骤四: 计算萤火虫 i 的邻居萤火虫集合 $N_i(t)$.

步骤五:利用式(2) 计算萤火虫 i 朝邻居集合内个体 j 的移动概率 $p_{ij}(t)$,并利用轮盘赌法在邻居集合中选择一个个体 j .

步骤六:利用式(3) 对所有萤火虫位置进行更新,如果 $N_i(t)$ 不为空,萤火虫的移动步长大小由式(5)、(6) 计算得到,若为空萤火虫则依据式(7),做 20 次试探性移动,选择最好的一个位置更新.

步骤七:利用式(4)对萤火虫的动态决策域半径进行更新.

步骤八:判断萤火虫算法是否达到最大迭代次数,如果达到,算法结束,输出最优值,如果未达到转向步骤二,继续运行算法.

4 算法仿真实验分析

4.1 实验平台

本实验运行环境为:处理器 CPU 是 i5 2.8 GHz,内存为 8 GB,操作系统是 windows 10,开发环境是 Matlab 2010b.

4.2 实验参数设置

算法的参数如下:萤火虫个体种群大小 n 是 100,迭代次数 $t = 200$, $p = 1$, $q = 0.1$ 荧光素挥发因子是 $\rho = 0.4$,荧光素更新率 $\gamma = 0.6$,萤火虫初始荧光素量 $I_0 = 5$,初始步长 $Step_0 = 0.1$,最大移动步长 $Step_{\max} = 1$,最小移动步长 $Step_{\min} = 0.01$,邻居集合阈值 $N_i = 6$,决策半径 $r_d = 2.048$,感知半径 $r_s = 2.048$. 原始 GSO 和改进算法的详细参数见表 1.

4.3 实验测试函数

为了验证算法的有效性本文采用了 6 个标准函数对 DSGSO 和原始 GSO 算法进行测试,所有的函

表 1 GSO 和 DSGSO 参数表

算法	ρ	γ	N_t	l_0	r_s	r_d	$Step_{\max}$	$Step_{\min}$	$Step_0$
DSGSO	0.4	0.6	6	5	2.048	2.048	1.0	0.001	—
GSO	0.4	0.6	6	5	2.048	2.048	—	—	0.03

数都是在指定目标区间内,找到最小值. 测试函数如下:

(1)sphere 函数

$$f_1(x)=\sum_i^n x_i^2$$
$$x_i \in [-10,10], i=1,\cdots 10$$

该函数的最小值是 0

(2)Becker and Lago

$$f_2(x)=(|x_1|-5)^2-(|x_2|+5)^2$$
$$x_1 \in [-10,10], x_2 \in [-10,10]$$

该函数的最小值是 25

(3)Schaffer F6

$$f_3(x)=0.5+\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1.0+0.001*(x_1^2+x_2^2)]^2}$$
$$x_1 \in [-10,10], x_2 \in [-10,10]$$

该函数的最小值是 0

(4)BR-Branin

$$f_4(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$$
$$x_1 \in [-10,10], x_2 \in [-10,10]$$

该函数的最小值是 -2

(5)RA-Rastrigin

$$f_5(x)=x_1^2+x_2^2-\cos 18x_1-\cos 18x_2$$
$$x_1 \in [-10,10], x_2 \in [-10,10]$$

该函数的最小值是 0

(6)Rosenbrock

$$f_6(x)=100(x_2-x_1^2)^2+(x_1-1)^2$$
$$x_1 \in [-10,10], x_2 \in [-10,10]$$

该函数的最小值是 0

(7)Modified_Rosenbrock

$$f_7(x)=100(x_2-x_1^2)^2+(6.4(x_2-0.5)^2-x_1-0.6)^2$$
$$x_1 \in [-5,5], x_2 \in [-5,5]$$

该函数的最小值是 0

(8)Exponential 函数

$$f_8(x)=-e^{-0.5\sum_{i=1}^n x_i^2}$$
$$x_i \in [-5,5], i=1,\cdots 10$$

该函数的最小值是 -1

4.4 仿真实验

下面对 $f_1(x) \sim f_8(x)$ 这 8 个标准测试函数^[2-4] 分别进行 20 次独立重复实验,分别取其最优值、最差值、平均值和标准偏差. 实验结果中的平均值、最差值、最差值反映了算法的优劣性,标准偏差反映了算法的稳定性,20 次独立重复实验结果见表 2. 为了证明本算法的有效性,本文还引进了参考文献自适应步长萤火虫优化算法^[2] 作为对比算法,该算法是基于荧光因子的自适应调整步长的策略.

表 2 本文算法 DSGSO 和 ASGSO,GSO 算法性能对比

测试函数	算法	最优值	最差值	平均值	标准偏差	理论极值
$f_1(x)$	GSO	23.391 235 01	39.881 367 16	31.627 092 01	4.628 342 712	0
	ASGSO	0.001 945 447	1.566 734 381	0.120 039 904	0.361 982 998	
	DSGSO	5.01E-05	0.090 278 353	0.021 740 039	0.028 251 629	
$f_2(x)$	GSO	25.000 087 93	26.863 151 37	25.523 690 57	0.7188 495 94	25
	ASGSO	25.000 000 515	25.043 052 83	25.002 235 44	0.009 608 768	
	DSGSO	25.000 000 004	25.000 000 2	25.000 000 05	4.50E-08	
$f_3(x)$	GSO	1.17E-04	0.009 763 423	0.008 885 694	0.002 596 596	0
	ASGSO	1.08E-10	0.009 715 91	0.009 230 114	0.002 172 543	
	DSGSO	2.08E-11	5.28E-10	3.48E-10	4.35E-14	
$f_4(x)$	GSO	0.397 889 191	1.635 351 122	0.580 309 228	0.427 073 826	0.398
	ASGSO	0.397 917 554	0.397 887 358	0.397 889 457	7.10E-06	
	DSGSO	0.398 000 358	0.397 887 381	0.397 887 365	6.17E-09	

续表 2

测试函数	算法	最优值	最差值	平均值	标准偏差	理论极值
$f_5(x)$	GSO	-1.952 260 37	3.257 569	0.187 228 667	1.430 528 905	-2
	ASGSO	-1.969 724 911	1.848 616 414	1.943 137 649	0.049 608 293	
	DSGSO	-1.999 725 127	-1.969 723 96	1.969 724 775	3.40E-07	
$f_6(x)$	GSO	1.38E-05	0.174 447 024	0.019 808 971	0.043 286 174	0
	ASGSO	3.38E-10	1.51E-05	4.08E-09	3.40E-06	
	DSGSO	8.43E-11	1.51E-05	1.12E-08	2.25E-08	
$f_7(x)$	GSO	6.74E-05	1.51E-05	0.001 139 844	9.98E-04	0
	ASGSO	2.80E-09	1.51E-05	1.40E-07	2.03E-07	
	DSGSO	1.21E-10	1.51E-05	8.93E-08	1.39E-08	
$f_8(x)$	GSO	-6.84E-05	1.51E-05	-9.61E-06	1.39E-07	-1
	ASGSO	-0.999 430 563	-0.976 513 4	-0.984 727 405	7.39E-07	
	DSGSO	-0.999 996 132	0.992 062 14	-0.993 694 321	1.39E-07	

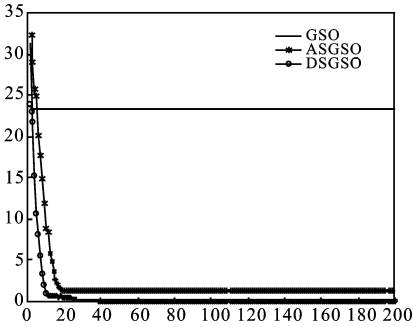


图 2 $f_1(x)$ 函数优化对比

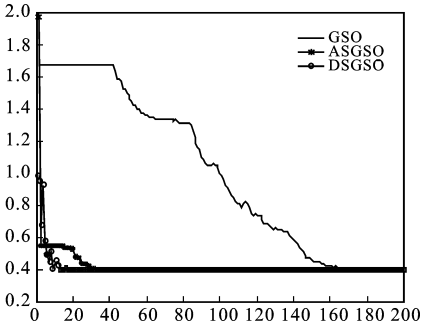


图 5 $f_4(x)$ 函数优化对比

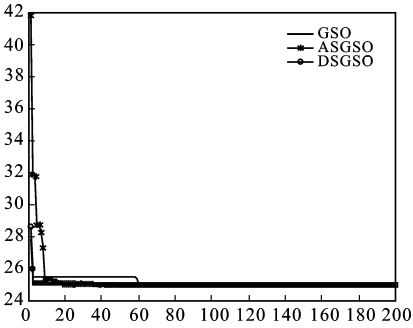


图 3 $f_2(x)$ 函数优化对比

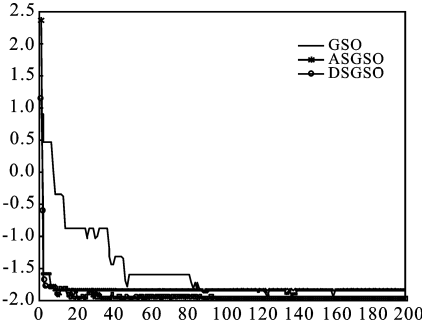


图 6 $f_5(x)$ 函数优化对比

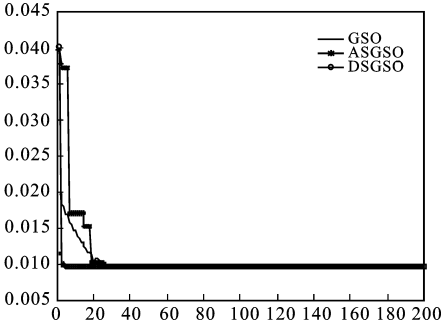


图 4 $f_3(x)$ 函数优化对比

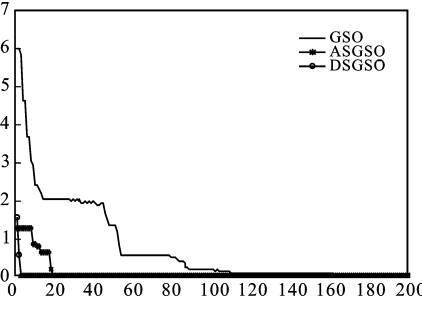
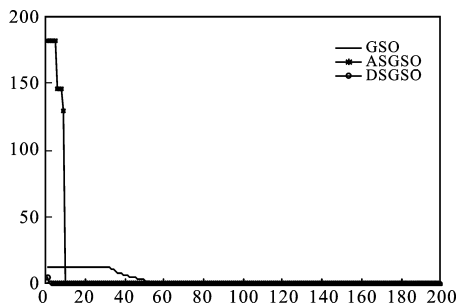
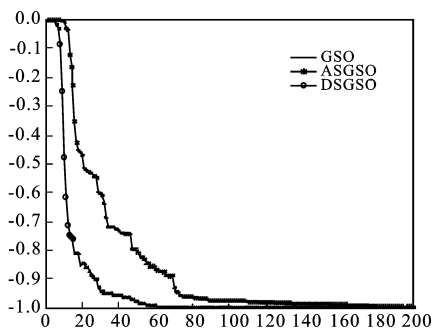


图 7 $f_6(x)$ 函数优化对比

图 8 $f_7(x)$ 函数优化对比图 9 $f_8(x)$ 函数优化对比

从实验结果可以看出,自适应步长的萤火虫算法和动态步长萤火虫优化算法比原始萤火虫优化算法搜索最优值的能力都要更好,寻优的最优值在目标搜索空间内也更加接近理论极值。对于 2 个高维函数 $f_1(x)$, $f_8(x)$ 原始萤火虫算法因为固定步长较小,移动能力较弱,无法求得最优值。对于 $f_2(x) - f_7(x)$,原始萤火虫算法虽然在 20 次重复试验中,可以找到接近理论极值,但是标准偏差有时候标新的偏大,最差值与理论极值偏差的较大。对比两个改进的算法,使用动态步长的萤火虫优化算法在寻优的过程中,要比参考文献自适应步长萤火虫优化算法^[5-7]更加接近理论极值,而且标准偏差表现的更好,算法十分稳定。依据图 2~9 来分析算法收敛性。对于 $f_1(x)$, $f_8(x)$ 函数,原始算法没有找到最优值,两个改进的算法对于 $f_1(x)$ 收敛速度基本相当,但是动态步长优化算法比自适应步长算法的收敛精度要更好,但是对于 $f_8(x)$,本文改进的算法第 60 次迭代就已经得到稳定的极值。参考文献算法大约在 150 次迭代之后得到一个稳定的极值,收敛速度较差。对于 $f_2(x) - f_7(x)$,除了 $f_3(x)$ 3 个算法收敛速度基本相当,其它函数,两个改进的算法收敛速度要快于原始算法,动态步长优化策略要强于自适应步长策略。

5 结束语

本文基于动态步长的思想,弥补了原始萤火虫算法固定步长的缺陷,萤火虫的移动步长随着算法的迭代次数增加,呈现非线性递减,最终保持着一个最小步长精细局部搜索。另外当邻居集合为空时,让萤火虫做试探性随机移动也避免了一部分萤火虫失去移动能力的问题。仿真实验中原始萤火虫算法几乎无法求得高维函数的最优值。改进后的算法无论是最优值、稳定性,还是收敛速度都强于原始算法,与参考文献的改进算法相比,有一定的优势。虽然改进后的算法在函数求解最优问题时表现出一定的优越性,但是在实际问题的性能怎么样还有待于实际验证。

参考文献:

- [1] Krishand 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] 欧阳喆, 周永权. 自适应步长萤火虫优化算法[J]. 计算机应用, 2011, 31(7): 1804-1807.
- [3] Deep K, Bansal J C Mean particle swarm optimization for function optimization [J]. International Studies, 2009, 1(1): 72-79.
- [4] 贺春华, 张湘伟. 竞选算法的参数设计与性能研究[J]. 计算机工程, 2010, 36(6): 201-203.
- [5] Simakov D, Caspi Y, Shechtman E, et al. Summarizing visual data using bidirectional similarity[C] //Proc of CVPR 08. Anchorage, Alaska, USA, 2008.
- [6] 郁书好, 苏守宝. 一种改进的变步长萤火虫优化算法[J]. 小型微型计算机系统, 2014, 35(6): 1396-1400.
- [7] 段艳明, 肖辉辉. 一种新的自适应步长果蝇优化算法[J]. 河南师范大学学报(自然科学版), 2016, 56(1): 161-168.

作者简介:

左仲亮 男, (1992-), 硕士研究生. 研究方向为个性化推荐, 群智能算法, 智能嵌入式硬件.

E-mail: 3041427120@qq.com.

郭星 男, (1980-), 男, 讲师. 研究方向为个性化推荐.

李炜 女, (1969-), 教授. 研究方向为软件工程, 智能数据处理.