

一种新颖的改进萤火虫算法

左仲亮, 郭 星, 李 炜

(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)对于多峰函数寻优精度不高和后期收敛速度较慢的问题, 为此, 本文针对性的提出了一种改进的动态步长自适应的萤火虫优化算法, 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. Matlab 实验仿真表明, 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; 自适应

中图分类号: TP393

文献标识码: A

文章编号: 1000—7180(2017)09—0015—05

An Original and Improved Swarm Optimization Alogorithm

ZUO Zhong-liang, GUO Xing, LI Wei

(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 and slow convergence. Thus, this paper comes up with an original and improved Dynamic and adaptive step optimization algorithm. 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. According to the simulation experiment, it shows that to some extent, the improved algorithm on covergence spend and precision are enhanced.

Key words: GSO; multi peak function; dynamic step; adaptive

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^[2-3]. 该算法具有操作简单, 鲁棒性强, 易于实现等优点. 但也存在显而易见的缺点, 容易早熟, 陷入局部极值点和震荡的缺点. 针对这些缺点, 本文提出一种新颖的改进萤火虫算法, 每一代萤火虫迭代之后, 都会得到单个个体的适应度, 对这些个体适应度加和求平均值, 通过和上

一代相比, 来动态调控萤火虫的移动步长. 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

在原始 GSO 算法中, 会把 N 只萤火虫个体随机的分布在一个 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 $[0, r_s]$) 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

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.

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(1):

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

公式中 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, 都会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, 萤火虫 j 的荧光素值必须必萤火虫 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:

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

在式(2)中, $j \in N_i(t)$, $N_i(t) = \{j: d_{i,j}(t) < r_d^i; l_i(t) < l_j(t)\}$, $d_{i,j}(t)$ 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:

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

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, 会动态的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. 依据公式(4) 进行更新:

$$r_i^d(t+1) = \min\{r_s, \max\{0, r_i^d(t) + \beta * (n_t - |N_i(t)|)\}\} \quad (4)$$

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_t 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

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, 若步长取的比较大, 则越容易搜索全局最优值, 但同时降低了最优值的精度. 若步长的保持的较小, 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. 因此本文针对这个问题, 在每次迭代之后, 求出整个萤火虫的平均适应度, 依据适应度去动态的调整移动步长的大小. 通过这个策略能够很好的调整全局寻优能力和寻优之间的平衡, 使算法的效率变高.

3.1 基于平均适应度的动态步长萤火虫优化算法

上述基本的萤火虫算法存在的不足, 本文采用公式(5-7) 来动态的改变萤火虫移动步长. 在算法迭代的第一次, 设定一个步长 S_0 , 在目标空间搜索. 第 t 代整个萤火虫的平均适应度 F_t 计算公式如下:

$$F_t = \frac{\left(\sum_{i=1}^N f_i(t) \right)}{N} \quad (5)$$

式中, N 代表萤火虫数量, $f_i(t)$ 代表第 i 只萤火虫在第 t 次迭代时对应的适应度. F_t 就代表在 t 次迭代时的平均适应度. 萤火虫平均适应度值得相对变化率用公式(6) 求得:

$$\text{rate} = \frac{F_{t-1} - F_t}{F_{t-1}}, \quad t \geq 2 \quad (6)$$

当相对变化率 rate 较大时, 表明目标搜索空间相对于步长来说较平滑和单调, 当前萤火虫的移动步长对搜索空间的探索效率较高. 因此适当的增大萤火虫的移动步长可以进一步提高算法的开采性能, 据此可以在一定程度上减少算法的迭代次数, 提

高搜索效率. 当计算得到的 rate 值较小时,表明此刻目标搜索空间相对于移动步长来说比较复杂,找到更优解的概率较低,因此适当的缩小移动步长来增加算法的局部探索能力,提高算法的搜索精度. 综上,本文提出的步长调整公式如下:

Step_{t+1} =
$$\begin{cases} \min[\text{Step}_t * (1 + p * \ln(1 + \text{rate})), \text{Step}_{\max}], & \text{rate} > 1 \\ \text{Step}_t, & 0.01 \leq \text{rate} \leq 1 \\ \max[\text{Step}_t / (1 + q * e^{-\text{rate}}), \text{Step}_{\min}], & \text{rate} < 0.01 \end{cases}$$

(7)

公式(7) 中 Step_t 为萤火虫在第 *t* 次迭代的移动步长. *p* 和 *q* 是移动步长变化幅度调节因子. 因为在迭代的过程中 rate 的值变化幅度较大. 本文在移动步长增大的过程中使用自然对数函数,在移动步长需要减小时使用 *e* 作为指数. 当 rate 的值较大时,萤火虫算法对应的移动步长也会相应增大,趋向最优值的速度的速度加快,全局寻优能力增强. 当 rate 值较小时,表示算法接近最优值,对应萤火虫的移动步长就会变小,局部搜索能力变强.

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 ISGSO.
- 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 Step_{max} 和最小移动步长 Step_{min}, 迭代次数 iteration, 和 $\rho, \gamma, \beta, \eta, s, \iota_0, r_0, \rho$ 等参数.
- 步骤二 使用式(1) 将萤火虫 *i* 在第 *t* 次迭代生成的位置 *x_i(t)* 所对应的目标函数值 *J(x_i(t+1))*,

- 转换成萤火虫的荧光素值 *l_i(t)*.
- 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 利用公式(3)对所有萤火虫位置进行更新,萤火虫的移动步长大小由公式(5-7)计算得到.
- 步骤七 利用公式(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 120, 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 *l₀* = 5,初始步长 Step₀ = 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_t* = 6,决策半径 *r_d* = 2.048,感知半径 *r_s* = 2.048. 原始 GSO 和改进算法的参数见表 1.

表 1 GSO 和 ISGSO 参数表

算法	ρ	γ	<i>N_t</i>	<i>l₀</i>	<i>r_s</i>	<i>r_d</i>	Step _{max}	Step _{min}	Step ₀
ISGSO	0.4	0.6	6	5	2.048	2.048	1	0.01	0.05
GSO	0.4	0.6	6	5	2.048	2.048	—	—	0.03

4.3 实验测试函数

为了验证 ISGSO 算法的有效性本文采用了 6 个标准函数对改进的 GSO 和原始 GSO 算法进行测试,所有的函数都是在指定目标区间内,找到最小值. 测试函数如下:

- (1)Becker and Lago
- $$f_1(x) = (|x_1| - 5)^2 - (|x_2| + 5)^2$$
- $$x_1 \in [-10, 10], x_2 \in [-10, 10]$$
- 该函数的最小值是 25
- (2)Schaffer F6
- $$f_2(x) = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1.0 + 0.001 * (x_1^2 + x_2^2)]^2}$$
- $$x_1 \in [-10, 10], x_2 \in [-10, 10]$$

- 该函数的最小值是 0
- (3)BR-Branin
- $$f_3(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$
- $$x_1 \in [-10, 10], x_2 \in [-10, 10]$$
- 该函数的最小值是 -2
- (4)RA-Rastrigin
- $$f_4(x) = x_1^2 + x_2^2 - \cos 18x_1 - \cos 18x_2$$
- $$x_1 \in [-10, 10], x_2 \in [-10, 10]$$
- 该函数的最小值是 0
- (5)Rosenbrock
- $$f_5(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

$x_1 \in [-10,10], x_2 \in [-10,10]$

该函数的最小值是 0

(6)Modified_Rosenbrock

$$f_6(x) = 100 (x_2 - x_1^2)^2 + (6.4 (x_2 - 0.5)^2 - x_1 - 0.6)^2$$

$x_1 \in [-5,5], x_2 \in [-5,5]$

该函数的最小值是 0

4.4 实验结果分析

对 $f_1(x) \sim f_6(x)$ 这 6 个标准测试函数^[4-5] 分别进行 20 次独立重复实验, 分别取其最优值, 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. 从表 2 中可以看出, 改进的萤火虫算法 ISGSO 无论是最优值, 平均值, 最差值, 和标准偏差都比原始 GSO 算法要优. 由此可得出结论, 改进的 ISGSO 算法比原始 GSO 算法寻优精度更高, 稳定性更好.

对于算法的收敛表现, 可以从图 1~图 6 得到的结果. 可以很容易的看出 ISGSO 算法比原始的 GSO 算法收敛的速度要快. 原始萤火虫的算法采用的是固定移动步长, 不能根据迭代的具体情况去改变搜索. ISGSO 算法利用平均适应度去感知搜索的结果的优劣, 去动态调整自己的移动步长.

表 2 算法最差值、最优值、平均值、标准差比较

函数	算法	最差值	最优值	平均值	标准差
$f_1(x)$	GSO	25.263 724 14	25.009 811 07	25.034 606 96	0.077 436 555
	ISGSO	25.010 657 07	25.000 286 8	25.002 751 65	0.002 643 536
$f_2(x)$	GSO	0.009 716 544	6.02E-04	0.008 298 145	0.003 463 292
	ISGSO	0.000 732 366	3.01E-05	0.000 323 289	0.000 166 126
$f_3(x)$	GSO	1.138 031 658	0.397 894 78	0.478 120 323	0.226 714 022
	ISGSO	0.400 182 767	0.398 005 711	0.398 460 663	6.18E-04
$f_4(x)$	GSO	1.666 8521 29	-1.968 191 65	-1.197 325 35	0.875 344 78
	ISGSO	-1.962 721 16	-2.002 721 16	-1.999 721 16	0.072 981 924
$f_5(x)$	GSO	0.010 088 131	9.71E-06	0.125 830 44	0.029 615 528
	ISGSO	0.005 825 646	1.40E-08	0.001 183 221	0.001 988 692
$f_6(x)$	GSO	0.042 026 514	2.57E-04	0.013 434 662	0.0120 762 24
	ISGSO	0.003 767 515	4.15E-05	0.001 360 17	0.001 106 641

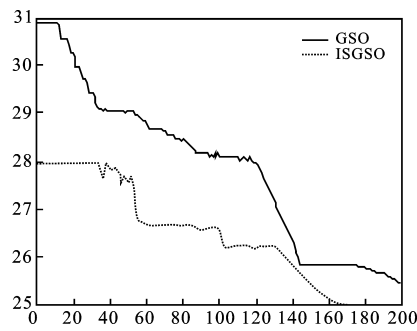


图 1 $f_1(x)$ 函数优化对比

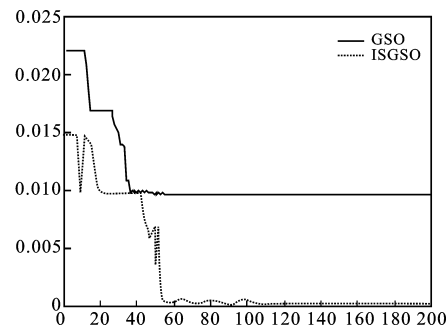


图 2 $f_2(x)$ 函数优化对比

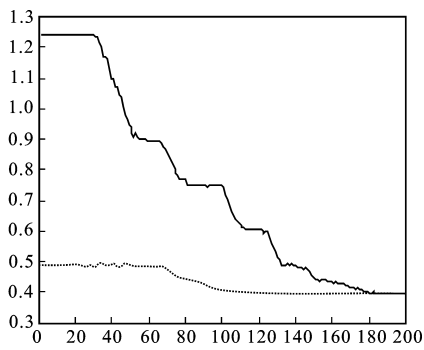


图 3 $f_3(x)$ 函数优化对比

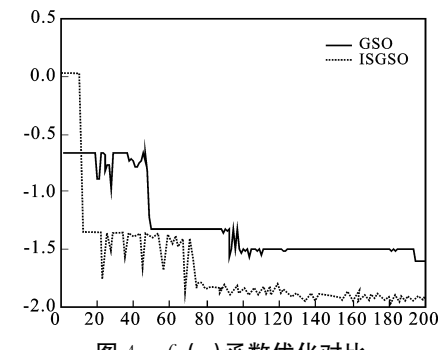


图 4 $f_4(x)$ 函数优化对比

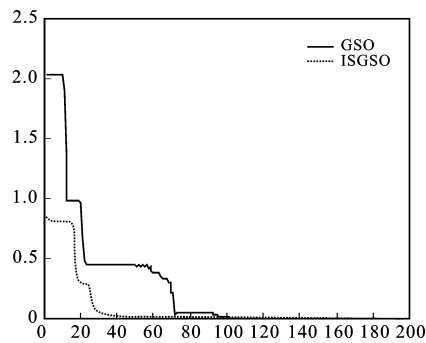


图 5 $f_5(x)$ 函数优化对比

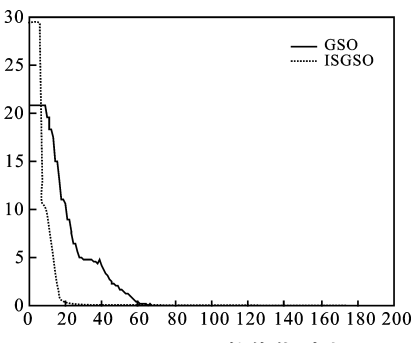


图 6 $f_6(x)$ 函数优化对比

5 结束语

本文基于平均适应度,动态调节萤火虫的移动步长的策略^[6-7].当相对变化率值较大时,保持着一个较大的步长,提升算法的全局搜索能力.当相对变化率较小时,适当减小步长,增强算法的局部寻优能力.仿真实验结果表明,改进的萤火虫算法 ISGSO 在寻优精度,收敛速度和稳定性上面都要优于 GSO 算法.对于算法的参数优化和算法在实际问题中的应用,也是下一步要研究的方向.

参考文献:

[1] Krishand K N,Ghose D. Glowworm swarm optimisation for simultaneous capture of mutiple local optima of multimodal funcations [J]. Swarm Intelligence,2009,3 (2):87-124.

[2] 曾冰,李明富,张翼. 基于改进萤火虫算法的装配序列规划方法[J]. 计算机集成制造系统, 2014, 20(4): 799-866.

[3] 田梦楚,陈志敏. 萤火虫算法智能优化粒子滤波[J].

自动化学报,2016,42(1):89-97.

[4] 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.

[5] Deep K, Bansal J C. Mean particle swarm optimization for function optimization [J]. International Studies, 2009,1(1):72-79.

[6] Simakov D, Caspi Y,Shechtman E,et al. Summarizing visual data using bidirectionnal similarity[C]//Proc of CVPR 08. Anchorage, Alaska, USA:[s. n],2008.

[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. 研究方向为群智能算法研究、智能嵌入式硬件、大数据.
E-mail:1039670791@qq. com.

郭 星 男,(1980-),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.