文章编号:1000-2367(2016)01-0161-08 DOI:10.16366/j.cnki.1000-2367.2016.01.029

一种新的自适应步长果蝇优化算法

段艳明[1], 肖辉辉[1,2]

(1.河池学院 计算机与信息工程学院, 江西 宜州 546300; 2.江西财经大学 信息管理学院, 南昌 330013)

摘 要:针对基本果蝇优化算法(FOA)易陷入局部最优、寻优精度低和后期收敛速度慢的问题,提出了一种自适应步长果蝇优化算法(ASFOA).该算法在运行过程中根据上一代最优味道浓度判断值和当前迭代次数来自适应调整进化移动步长,使算法在初期的步长大而避免种群个体陷入局部最优,到后期果蝇移动的步长变小而获得更高的收敛精度解,并加快收敛速度.通过6个标准测试函数对改进算法进行仿真测试,结果表明ASFOA算法具有更好的全局搜索能力,其收敛精度、收敛速度均比FOA算法及参考文献中其他改进果蝇优化算法有较大的提高.

关键词:自适应;果蝇优化算法;收敛速度;味道浓度

中图分类号:TP301.6 **文献标志码**:A

果蝇优化算法(Fruit Fly Optimization Algorithm, FOA)是台湾潘文超博士在2011年6月从果蝇觅食行为中得到启发而提出的.FOA是一种寻求全局优化的群智能算法[1-3].该算法可应用于科学计算和工程领域,也可和其他的智能算法融合用于数据挖掘.现已将其成功应用于Z-SCORE模型系数的微调、函数极值的求解、灰色神经网络参数的优化等[4].FOA算法相对其他群智能算法(遗传算法,蚁群算法,粒子群算法,免疫算法,鱼群算法等),简单易理解,程序易实现,运行时间少,需调整的参数少[4-5],这样就极大地提高了FOA算法的效率与应用.但是,FOA算法与遗传算法、粒子群算法等全局优化算法类似,极易陷入局部最优,而导致收敛精度低,后期的收敛速度慢,尤其对于多局部极值、高维的较复杂优化问题.另外,由于FOA算法提出时间较晚,目前国内外对该算法的研究还处于初级阶段,算法理论尚未成熟,研究成果也较少,因此对FOA的研究迫切而有价值.

针对FOA的缺陷,韩俊英等人提出了自适应混沌果蝇优化算法(ACFOA)[6],该算法应用混沌机制对果蝇进行全局寻优.韩俊英等人又提出了自适应调整参数的果蝇优化算法(FOAAP)[7],在每个进化代由逆向云发生器自适应调整果蝇个体的方向与距离.韩俊英等人又提出了动态双子群协同进化果蝇优化算法(DDSCFOA)[8],通过动态划分种群为先进子群和后进子群,分别对其采用混沌和基本果蝇算法进行平衡局部和全局搜索能力.刘成忠等人提出了基于细菌迁徙的自适应果蝇优化算法(AFOABM)[9],该算法在运行过程中根据进化停滞步数的大小自适应地引入细菌迁徙操作.张前图等人提出了具有Lévy飞行特征的双子群果蝇优化算法(LFOA)[10],引入Lévy飞行策略提高局部搜索能力.以上对果蝇算法的改进都在一定程度上使基本FOA跳出局部极值,提高了算法寻优能力,但仍未使算法完全避免陷入局部极值,其收敛精度、稳定性、收敛速度等方面仍存在不足.为此,本文提出自适应步长果蝇优化算法(ASFOA),该算法在优化过程中不再采用固定的步长,而是根据上一代最优味道浓度判断值和当前迭代次数自适应地调整果蝇个体的进化步长.因此,ASFOA能更有效地跳出局部极值,提高果蝇优化算法的收敛速度和寻优精度.

收稿日期:2015-01-06;**修回日期**:2015-09-16.

基金项目:国家自然科学基金(61173146);广西高校科研项目(KY2015LX332; KY2015LX334);校级项目(XJ2015QN003);江西省研究生创新项目(YC2015-B054);河池学院计算机网络与软件新技术重点实验室项目(院科研2013(3)号).

第1作者简介:段艳明(1978-),女,江西永新人,河池学院讲师,研究方向为智能计算,E-mail:yanhui0920@126.com.

通信作者:肖辉辉,男,副教授,博士生,E-mail:gxhcxyxzy@126.com.

1 果蝇优化算法

1.1 果蝇优化算法

果蝇利用自身在嗅觉与视觉上优于其他物种的特点,先利用嗅觉搜集飘浮在空气中的各种气味,飞近食物后,再利用敏锐的视觉发现食物与同伴聚集的位置,形成新的果蝇群体位置,再沿随机方向飞出,再往食物浓度高的果蝇位置聚合,不断循环反复,直到找到食物[1].具体算法步骤[1-3]如下:

1) 参数初始化.初始化群体规模 *sizepop*;最大迭代次数 maxgen;果蝇群体初始位置 X_a 和 Y_a.

2) 赋予果蝇个体利用嗅觉搜寻食物的随机方向与距离,式中 R_V 为搜索距离:

$$X_i = X_a + R_V, \quad Y_i = Y_a + R_V. \tag{1}$$

3) 由于无法得知食物位置,因此先估计与原点之间的距离 D_i,再计算味道浓度判断值 S_i,此值为距离之倒数:

$$D_i = \sqrt{X_i^2 + Y_i^2}, S_i = 1/D_i. \tag{2}$$

4) 将味道浓度判断值 S_i 代入味道浓度判断函数(适应度函数),求得果蝇个体的味道浓度

$$S_{m_i} = f(S_i). \tag{3}$$

5) 找出果蝇种群中味道浓度最佳的果蝇(最优个体):

$$[bestSmell \quad bestindex] = \min(S_{m_i}). \tag{4}$$

6) 记录并保留最佳味道浓度值 *bestSmell* 与其 X,Y 坐标.这时候果蝇群体利用视觉向该位置飞去:

$$Smellbest = bestSmell, X_a = X(bestindex), Y_a = Y(bestindex). \tag{5}$$

7) 进入迭代寻优,重复执行步骤 2)~步骤 5),并判断最佳味道浓度是否优于前一次迭代的最佳味道浓度,并且当前迭代次数小于最大迭代数 maxgen,若是执行步骤 6);否则,结束算法.

1.2 果蝇优化算法的局限性

果蝇优化算法的本质是将果蝇种群随机初始化,再将果蝇种群个体进行固定位移步长的位置更新,然后果蝇种群内所有果蝇个体复制为适应度最优的果蝇个体(飞向适应度最优的果蝇个体位置),再利用新位置上的果蝇种群继续更新位置,这与粒子群算法类似.因此算法存固有的不足:FOA 采用了向当前最优个体学习进行位置更新,若当前最优个体陷入局部极值,且没有有效的机制来跳出该局部极值,种群会停止进化,全局寻优失败;FOA 算法中果蝇群体在进化中是采用固定步长,则制约了算法的收敛性能和稳定性能,若固定步长取值过大,则易导致果蝇跳离最优解,稳定性差,后期易出现振荡等问题,反之,则又可能过早地陷入局部最优解,算法收敛慢,收敛精度低.因此,FOA 易陷入局部最优、收敛速度慢及收敛精度低,特别是求解多峰、高维的复杂问题.

2 自适应步长的果蝇优化算法(ASFOA)

通过对 FOA 算法存在的问题的分析可知,影响其收敛速度和收敛精度的主要因素为初始位置、种群大小和进化迭代步长.而初始位置由函数的自变量决定,种群规模的大小也影响着运行时间.因此,如何选取合适的进化迭代步长是避免算法陷入局部最优,提高收敛速度及收敛精度的关键.鉴于此,本文提出了一种自适应步长的果蝇优化算法 ASFOA,该算法中果蝇的进化步长不再固定,自适应地根据上一代的最优味道浓度和当前迭代次数来调整步长大小.

2.1 自适应步长调整策略

对 FOA 算法的式(2)和式(3)进行分析,有 $X_i = \pm\sqrt{S_i^2 - Y_i^2}$,考虑到 Y_i 的影响,假设 $X_i = Y_i$,则有 $X_i = \frac{1}{\sqrt{2}} \times S$,故果蝇的位置$(X_i, Y_i)$受味道浓度判断值 S_i 的影响.同时考虑当前进化程度即当前迭代次数与最大迭代次数之间的关系,对果蝇的进化移动步长 h_i 设置

$$h_i = \frac{m}{bestS_{i-1}} \times \exp\left(-k \times \left(\frac{t}{t_{\max}}\right)^p\right) + h_{\min}. \tag{6}$$

其中：m 为调节因子其取值范围在 0～1 之间；$bestS_{i-1}$ 是上一代的群体最优味道浓度判断值；k 为限制因子，其取值范围在 0～1 之间；t 为当前迭代次数；$t_{\max}$ 为设置的最大迭代次数；$h_{\min}$ 为步长的最小值；p 为大于 1 的整数，可视具体情况取值范围为[1,30].

则，果蝇个体的搜索距离 R_V 为：

$$R_{V_i} = h_i \times (2 \times rand() - 1). \tag{7}$$

果蝇个体利用嗅觉搜寻食物的随机方向与距离更改为：

$$X_i = X_a + R_{V_i}, Y_i = Y_a + R_{V_i}. \tag{8}$$

这样，自适应步长 h_i 既考虑了上一代的最优味道浓度判断值，又考虑了迭代的进化而得到的收敛，使果蝇的移动步长随着迭代的进行而自适应改变，在算法初期保持一个较大的步长，以避免算法过早陷入局部最优，到迭代后期，移动步长自适应减小，从而使算法在后期快速收敛，提高收敛速度，并获得高精度的全局最优解.

2.2 ASFOA 算法的实现步骤

ASFOA 算法以 FOA 为主体流程，利用自适应调整的变步长来更新果蝇位置. ASFOA 算法的流程图如图 1 所示. 其中，*Smellbest* 为果蝇群体全局最佳味道浓度，X_a 和 Y_a 为全局最佳味道浓度的个体坐标，bestSmell 为果蝇群体在利用变步长更新位置后的全局最佳味道浓度. 在 ASFOA 迭代中，果蝇的变步长由(6) 式和(7) 式完成，果蝇个体位置的更新由(8) 式执行.

因此，ASFOA 算法中果蝇的移动步长也不再固定，而是根据上一代的最优味道浓度判断值和当前迭代次数而自适应调整. 自适应步长的果蝇优化算法在迭代的前期，种群的步长较大，使果蝇在大范围内进行粗搜索，从而达到更快地搜索到全局最优的范围；而到迭代的后期，即当趋于收敛的时候，种群步长变小，算法具有更强的局部搜索能力和更快的收敛速度，从而克服了 FOA 算法收敛精度不高、容易陷入局部最优、后期收敛速度慢等缺点.

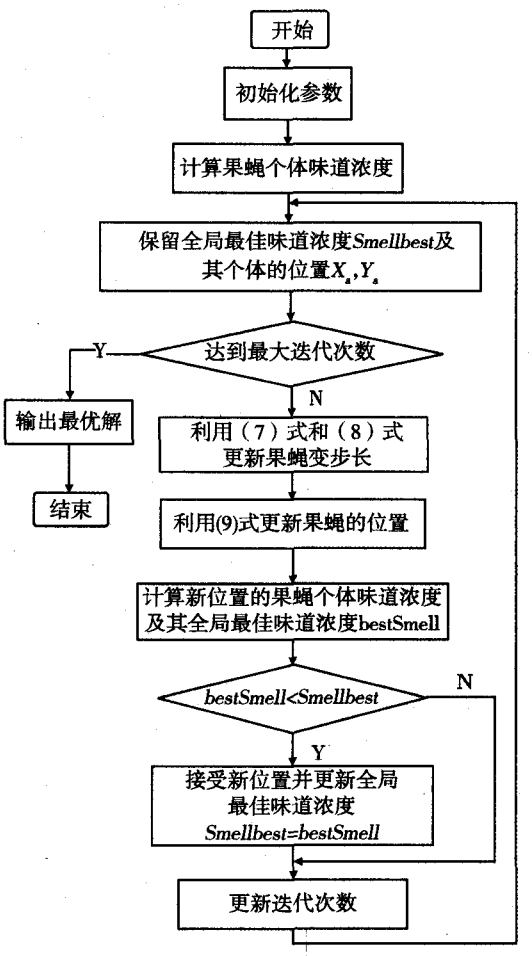

图1 ASFOA算法流程图

3 ASFOA 仿真实验及结果分析

3.1 标准测试函数

为了验证本文提出的 ASFOA 算法的收敛速度、寻优精度等性能优于 FOA 算法及其他群智能算法，选用 6 个常用于优化算法比较的基准函数进行仿真实验.

测试函数[6-9] 如下：

1) $f_1(x) = \sum_{i=1}^{n} x_i^2, x_i \in [-100,100], i = 1,2,\cdots,n.$

该函数是一个单峰函数，在 $x^* = (0,0,\cdots,0)$ 处取得全局最小值 $f_{\min}(x) = 0.$

2) $f_2(x) = \sum_{i=1}^{n} [x_i^2 + 10\cos(2\pi x_i) + 10], x_i \in [-5.12,5.12], i = 1,2,\cdots,n.$

该函数是一个多峰函数，在 $x^* = (0,0,\cdots,0)$ 处取得全局最小值 $f_{\min}(x) = 0.$

3) $f_3(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_i^n x_i^2}\right) - \exp\left(\frac{1}{n}\sum_i^n \cos(2\pi x_i)\right) + 20 + e, x_i \in [-32.768,32.768],$ $i = 1,2,\cdots,n.$

该函数是一个多峰函数,在 $x^* = (0,0,\cdots,0)$ 处取得全局最小值 $f_{\min}(x) = 0$.

4) $f_4(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right), x_i \in [-600,600], i = 1,2,\cdots,n.$

该函数是一个多峰函数,在 $x^* = (0,0,\cdots,0)$ 处取得全局最小值 $f_{\min}(x) = 0$.

5) $f_5(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2], x_i \in [-2.048,2.048], i = 1,2,\cdots,n.$

该函数是一个单峰函数,在 $x^* = (0,0,\cdots,0)$ 处取得全局最小值 $f_{\min}(x) = 0$.

6) $f_6(x) = \frac{\sin^2\sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}, x_i \in [-100,100], i = 1,2,\cdots,n.$

该函数的是一个二维多峰函数,在 $x^* = (0,0)$ 处取得全局最小值 $f_{\min}(x) = -1$.

3.2 实验结果及分析

为了比较和突出 ASFOA 算法的性能,ASFOA 和 FOA,PSO,BA 的群体规模 $sizepop=30$,维数 $D=30$(函数 f_6 的维数 $D=2$),ASFOA 中的步长调节因子 $m=0.8$,限制因子 $k=0.2$,$p=5$;BA 算法参数:$A=0.25$,$r=0.5$,$alf=0.95$,$BAma=0.05$;PSO 算法参数:$c_1=c_2=2$,$w=0.9$,$v_{\max}$(最大速度)$=0.5$.

本文的性能评估方法为:1)固定迭代次数,测试 3 种算法 ASFOA,FOA 和 PSO 的寻优性能;2) 测试 ASFOA 算法在高维函数上的优化性能;3)与参考文献改进算法对比,测试本文算法的寻优精度和鲁棒性,以进一步验证 ASFOA 算法的寻优精度.4)ASFOA 算法和 FOA 算法在不同高维情况下的运行时间比较,以证明 ASFOA 的算法复杂度.

3.2.1 固定迭代次数的收敛精度和收敛速度

6 个测试函数的进化迭代次数固定为 300,分别采用基本 FOA,ASFOA 和 PSO 3 个算法对其进行求解,为了防止算法的偶然性带来的误差,对每个测试函数独立运行 20 次,并对 20 次得到的最优 *Smellbest* 值进行最优值、最差值、优化平均值和标准方差的运算,其实验结果如表 1 所示.其中,寻优成功率是指迭代过程中最优值达到函数的理论最优值或指定的目标精度的次数与总迭代次数的比,f_1 的目标精度为 e^{-5},f_3 的目标精度为 e^{-1},f_5 的目标精度为 28.8,f_2 和 f_4 的目标精度即为理论最优值 0.从表 1 可以看出:相比 FOA,PSO 算法,本文提出的 ASFOA 算法都能找到或更接近理论最优值,其求解得到的最优值、最差值、优化均值和标准方差均优于 FOA 算法以及 PSO 算法,说明 ASFOA 搜索最优值的能力更强,解的精度更高,具有更好的鲁棒性和稳定性.1)对于函数 f_2,f_4 和 f_6,ASFOA 能找到理论最优值 0,并且其最差值、优化均值均达到理论值 0,标准方差都为 0,而 FOA 和 PSO 的性能精度很低,说明 ASFOA 能很好地跳出局部极值,搜索到全局最优值,有着更高的寻优精度及更强的稳定性和鲁棒性.2)对函数 f_1,ASFOA 的最优值的精度为 e^{-308},ASFOA 的优化均值的精度为 e^{-305},均比 FOA 高 305 个数量级,比 PSO 高 307 个数量级,ASFOA 的标准方差达到 0.3) 对于函数 f_3,ASFOA 的最优值、最差值和优化均值精度均为 e^{-16},比 FOA 高 15 个数量级,比 PSO 高 16 个数量级;ASFOA 的标准方差为 0,而其他 FOA,PSO 算法的标准方差都较大,验证 ASFOA 有着更强的稳定性和鲁棒性.4)对于函数 f_5,ASFOA 的最优值、最差值、优化均值和标准方差较其他 3 个算法都要小,说明 ASFOA 算法性能最好.5)对于函数 $f_1 \sim f_4$ 及 f_6,ASFOA 都 100%的寻优成功率收敛到理论最优值或指定目标精度,而 FOA 的寻优成功率都为 0%,对于函数 f_5,ASFOA 的寻优成功率也远高于 FOA 的寻优成功率.

为了直观地反映出改进算法的寻优效果,图 2 给出了 6 个测试函数的收敛曲线图(为了方便收敛曲线的显示和观察,对部分函数的目标函数值取以 10 为底的对数),形象对比了 ASFOA,FOA 和 PSO 算法的适应度值的迭代下降过程和全局最优解的收敛速度.由图 2 的收敛曲线可直观地观察到:1)对测试的 6 个函数,与 FOA,PSO 算法相比,本文提出的 ASFOA 算法具有更好的寻优能力,其收敛速度、优化精度明显优于 FOA,PSO 算法.2)对于多峰函数 f_2,f_4,f_6,ASFOA 能很快收敛到理论最优值;对于其他多峰函数,ASFOA 的收敛速度同样远快于 FOA 和 PSO 算法;对于单峰函数 f_5 亦以很快地速度收敛到较 FOA、PSO 算法更优的值.

表 1　ASFOA 与 FOA、PSO 算法的性能比较

函数	维数	算法	最优值	优化均值	最差值	标准方差	寻优成功率/%
		PSO	0.3853	2.2060	6.4356	1.3987	0
f_1	30	FOA	0.0014	0.0017	0.0019	1.5842e-04	0
		ASFOA	1.7019e-308	5.4958e-305	5.3558e-304	0	100
		PSO	32.0838	42.1958	65.6796	9.2256	0
f_2	30	FOA	9.8808	93.9135	173.0757	45.1721	0
		ASFOA	0	0	0	0	100
		PSO	4.1063	5.6659	7.8572	1.3732	0
f_3	30	FOA	0.0508	0.0532	0.0581	0.0018	0
		ASFOA	8.8818e-16	8.8818e-16	8.8818e-16	0	100
		PSO	337.2854	374.7780	448.6702	26.6403	0
f_4	30	FOA	1.2630e-07	1.8195e-07	2.4690e-07	3.3793e-08	0
		ASFOA	0	0	0	0	100
		PSO	36.7059	54.0863	131.3123	22.4038	0
f_5	30	FOA	28.4396	32.7822	41.4198	2.5746	48
		ASFOA	26.3904	27.1517	27.9240	0.3366	92
		PSO	−0.999 999 995 65	−0.993 679 714 27	−0.993 679 714 27	0.0047	0
f_6	2	FOA	−0.999 994 731 50	−0.999 998 233 88	−0.999 989 764 98	3.1571e-06	0
		ASFOA	1	1	1	0	100

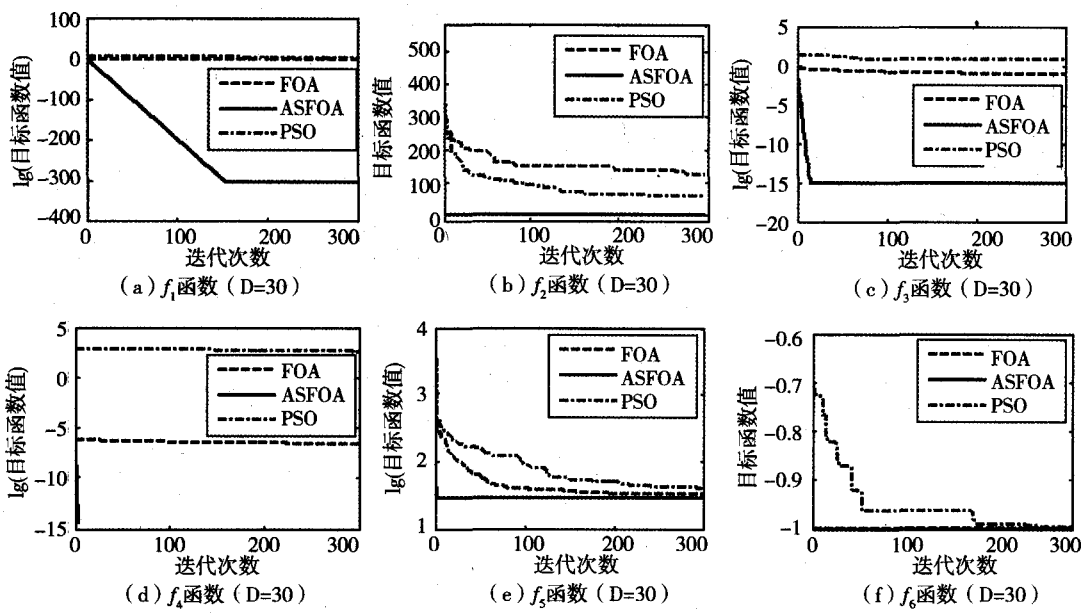

图2 3种算法在函数 f_1~f_6 上的收敛曲线

3.2.2　ASFOA 算法在高维函数上的性能优化比较

大部分智能优化算法均存在易陷入局部最优的缺陷，而导致算法在后期的收敛速度变慢、收敛精度低，尤其对于高维、多峰的复杂优化问题. 为了突出本文算法 ASFOA 在高维函数上的优势性能，进行以下两种情况比较：1)将 ASFOA 和 FOA、粒子群优化算法 PSO、蝙蝠算法 BA 及文献[10]中的遗传算法 GA 和细菌觅食优化算法 BFO 在 50 维上进行比较. 2)比较 ASFOA 和 FOA 在高维函数上随函数维数的增加其优化性能的变化. 本小节具体参数设置为：$Sizepop=15$，$Maxgen=800$；PSO 中的 $c1=c2=2$，$w=0.9$，v_{max}(最大速度)$=0.5$；BA 中的 $A=0.25$，$r=0.5$，$alf=0.95$，$BAma=0.05$.

第一种情况的测试结果如表 2 所示，表 2 中“—”表示文献中没有报道. 从表 2 可以看出，ASFOA 优化均值和标准方差显著优于 FOA，PSO，BA 和参考文献[11]的 GA，BFO. 对于函数 f_2 和 f_4，ASFOA 的优化

均值都收敛到全局理论最优值 0,且标准方差也都为 0;对于函数 f_3,ASFOA 的优化均值达到 e^{-16},标准方差为 0;对于函数 f_5,优化均值和标准方差也是最优的.说明 ASFOA 在高维函数上依然具有较高的优化性能,其在高维上的优化性能高于 FOA,PSO,BA 和参考文献[11]的 GA,BFO.

表 2 算法在 50 维函数上的性能比较

函数	性能	算法					
		ASFOA	FOA	PSO	BA	GA[11]	BFO[11]
f_2	优化均值	0	201.1077	54.8291	7.2802	1.505	1.028
	标准方差	0	69.4245	15.8343	32.5578	1.15	0.008
f_3	优化均值	8.8816e-16	0.0657	5.8688	0.9699	50.481	50.64
	标准方差	0	0.0016	0.8867	4.3376	35.48	0.23
f_4	优化均值	0	1.5389e-07	389.7100	15.7870	66.791	3.73
	标准方差	0	1.9073e-08	30.8355	70.6015	11.57	0.36
f_5	优化均值	28.6130	54.6212	43.6726	32.4126	—	—
	标准方差	0.0374	3.3582	14.3131	5.5653	—	—

第 2 种情况是对高维函数在不同高维条件下用 ASFOA 和 FOA 算法独立运行 20 次,函数维数从 60～160 维递增 20 维变化,以其优化均值及其平均变化率为评价指标,测试本文算法 ASFOA 相对于 FOA 的优化性能随函数维数的增加而变化的情况,结果如表 3 所示.其中,平均变化率$=\sum_{i=1}^{5}$((后一次维数的优化均值 V_{i+1}－前一次维数的优化均值 V_i)/前一次维数的优化均值 V_i)/5.从表 3 可以看出,随着函数维数的增加,FOA 的优化性能降低,即优化均值逐渐远离理论最优均值 0.而 ASFOA 的优化性能要好,对于函数 f_2～f_4,ASFOA 的优化均值随着维数的增加保持不变,特别对于函数 f_2 和 f_4,优化均值都保持在理论全局最优值.另外,对于函数 f_2～f_4,ASFOA 的平均变化率都为 0,而 FOA 的平均变化率都较大;对于函数 f_5,ASFOA 的优化均值随着维数的增加只增加较小值,ASFOA 的平均变化率也低于 FOA 的平均变化率.这说明随着维数的增加,ASFOA 的优化性能基本不降低,即不会随着复杂高维多极值函数的维数的增大而陷入"维灾难",相对于 FOA 更能突出其优势.为了更直观地对比 ASFOA 和 FOA 在高维情况下的性能,图 3～图 6 是这两种算法在不同维数下测试 f_2～f_5 的优化均值的曲线图.从图 3～图 6 可以看出,对所有测试函数,FOA 的优化均值随着维数的增加而单调增加,且增加的幅度较大.而 ASFOA 的优化均值在 f_2～f_4 中保持不变,在 f_5 中 ASFOA 的优化均值随维数的增加幅度小于 FOA.因此,采用自适应步长能使 FOA 跳出局部最优,在高维的多峰函数的优化中,寻优效果同样要好,进一步验证了本文算法的可行性和有效性.

表 3 多峰函数在不同高维上的优化均值比较

函数	算法	维数						平均变化率/%
		6	8	10	12	14	16	
f_2	FOA	2 203 389	3 109 035	3 930 317	4 300 683	5 655 576	5 691 227	0.1905
	ASFOA	0	0	0	0	0	0	0
f_3	FOA	0.0533	0.600	0.655	0.714	0.752	0.800	0.0849
	ASFOA	88 818e-16	8.8818e-16	8.8818e-16	8.8818e-16	8.8818e-16	8.8818e-16	0
f_4	FOA	13 698e-07	16 630e-07	17 744e-07	21 181e-07	24 299e-07	28 001e-07	0.1549
	ASFOA	0	0	0	0	0	0	0
f_5	FOA	645 125	84.4256	106.3624	143.4271	150.3516	1 720 804	0.2219
	ASFOA	58.5638	78.4853	98.4489	18.3751	138.3626	146.4076	0.2060

3.2.3 ASFOA 算法与参考文献算法的优化性能比较

为了进一步对比 ASFOA 算法的优势,将本文算法 ASFOA 与参考文献中改进的果蝇优化算法(ACFOA[6]、DDSCFOA[8]、AFOABM[9])和经典优化算法及其改进算法(AD-PSO[13]、ASCS[14])对相同测试函数的性能进行比较(参数设置同 3.2.1,ASFOA 的数据来自表 1),结果如表 4 所示,"－"表示参考文献算法没有报道.从表 4 中可以看出,ASFOA 优化性能(最优值、优化均值)较参考文献中的其他群智能优化算法提高很大.对于函数 f_2,f_4 和 f_6,ASFOA 能收敛到理论全局最小值;对于函数 f_1,f_3 和 f_5,ASFOA 的最优值和优化均值显著优于其他算法,最多相差 125 个数量级.因此,相对于参考文献算法,ASFOA 算法能更好地跳出局部极值,具有更高的收敛精度和更强的寻优能力.

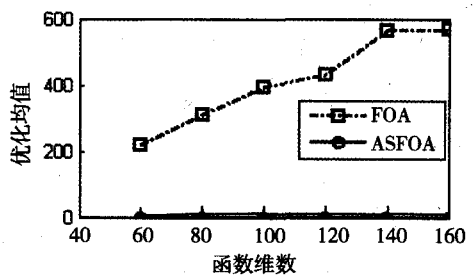

图3 f_2函数优化均值随函数维数变化曲线

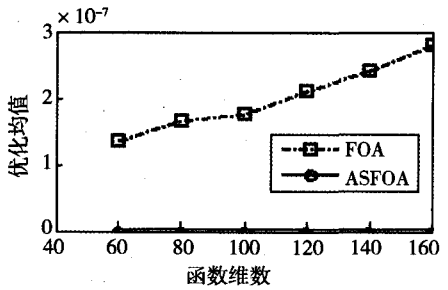

图5 f_4函数优化均值随函数维数变化曲线

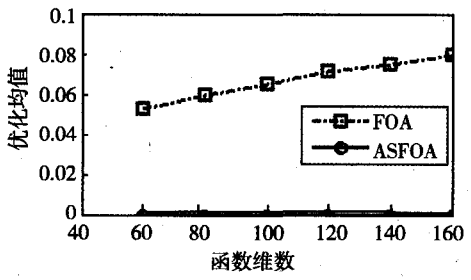

图4 f_3函数优化均值随函数维数变化曲线

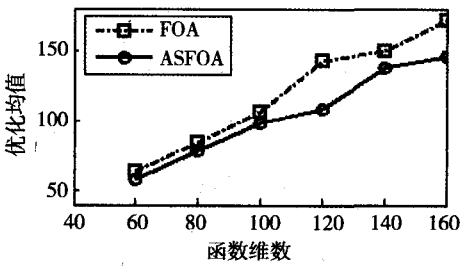

图6 f_5函数优化均值随函数维数变化曲线

表 4　ASFOA 算法与参考文献算法的最优值和优化均值比较

函数	性能	ASFOA	ACFOA[6]	DDSCFOA[8]	AFOABM[9]	OBLFOA[12]	AD-PSO[13]	ASCS[14]
f_1	最优值	1.7019e-308	3.7868e-21	—	2.7770e-6	4.1234e-6	—	3.1311e-08
	优化均值	5.4958e-305	3.8126e-21	3.8994017	3.1540e-6	4.6823e-6	2.1499e-25	1.0063e-06
f_2	最优值	0	1.7764e-15	—	9.4585e-6	3.4176e-5	—	9.0260
	优化均值	0	3.1086e-15	1.5143e-11	9.7825e-6	1.99e-2	1.7054+00	17.9013
f_3	最优值	8.8818e-16	1.6844	—	1.6845	1.6847	—	—
	优化均值	8.8818e-16	1.6844	—	1.6845	1.6847	—	—
f_4	最优值	0	0	—	5.3239e-8	9.2797e-8	—	1.0968
	优化均值	0	0	6.8122e-12	1.0355e-7	1.8151e-7	—	1.2444
f_5	最优值	26.3904	28.7070	—	28.5310	28.7070		2.4598e+03
	优化均值	27.1517	28.7327	28.7161	28.6839	28.7070	3.1554e+03	7.5411e+03
f_6	最优值	−1	−1.0000	—	1.0403e-7	−1.0000	—	2.6769e-07
	优化均值	−1	−09999	0	1.1596e-7	−1.0000	1.2548e-02	0.0019

3.2.4　算法复杂度的分析

改进算法是否有效、可行，除了性能上有较大的提高，算法的时间复杂度应该比较低，相对于原算法，运行的时间应该不能过长. 而算法的时间复杂度又跟算法中基本控制语句的执行次数有关，故测试函数的维数影响着算法的时间复杂度. 由于篇幅限制，本小节只选用测试函数中的多峰函数 f_2，f_3 和 f_4 3 个函数来对本文算法的时间复杂度进行测试并分析，设置种群数 $sizepop=30$，进化迭代次数 maxgen＝300，独立运行 20 次，计算两种算法在不同高维数(100 维、150 维、200 维和 250 维)所需要的平均运行时间，结果如表 5 所示(D 为维数). 从表 5 的平均运行时间来看，虽然 ASFOA 算法对果蝇群体进行了自适应步长的扰动，但其平均运行时间比 FOA 算法的平均运行时间只略长一些，进而说明 ASFOA 算法的复杂度较低，是可行和有效的.

由以上实验结果及分析可知，本文提出的 ASFOA 算法总体上较 FOA 算法、文献[6－9，12]中改进的 FOA 算法，以及文献[11]中的 GA，PSO 算法具有更快的收敛速度，更好的全局搜索能力和较高的收敛精度. 同时，也验证了本文 ASFOA 算法在一定程度上避免了局部极值的现象，提高了全局优化能力.

表 5 $f_2 \sim f_4$ 的平均运行时间对比

函数	算法	平均运行时间/s			
		$D=100$	$D=150$	$D=200$	$D=250$
f_2	FOA	0.7758	0.8859	1.0133	1.1313
	ASFOA	0.8547	0.9477	1.0586	1.1711
f_3	FOA	0.7883	0.9023	1.0227	1.1508
	ASFOA	0.8820	1.0094	1.1070	1.2320
f_4	FOA	1.3953	1.5289	1.6211	1.7477
	ASFOA	1.4883	1.6242	1.7242	1.8852

4 结束语

本文针对 FOA 算法后期收敛速度慢、易陷入局部最优等缺陷,提出了一种自适应步长 ASFOA 算法,进入迭代后根据上一代最优味道浓度判断值和当前迭代次数来自适应调整果蝇移动的步长. 通过 6 个标准测试函数测试,仿真结果表明,ASFOA 算法的收敛速度、鲁棒性、寻优精度得到较大幅度地提高. 如何利用其他智能算法的优点来改进 FOA 算法,如何扩展 FOA 算法的应用领域,如利用 FOA 算法解决离散性问题;如何设置其他自适应的参数值,参数的设置是否存在统一等问题是今后进一步研究的内容和方向.

参 考 文 献

[1] PAN W. A new fruit fly optimizatin algorithm: taking the financial distress model as an example[J]. Knowledge-based Systems, 2012, 26:69-74.

[2] 潘文超. 果蝇最佳化演算法[M]. 台北:沧海书局,2011:10-12.

[3] 潘文超. 应用果蝇优化算法优化广义回归神经网络进行企业经营绩效评估[J]. 太原理工大学学报:社会科学版,2011,29(4):1-5.

[4] 王 欣,杜 康,秦 斌,等. 基于果蝇优化算法的 LSSVR 干燥速率建模[J]. 控制工程,2013,19(4):630-633.

[5] 周爱国,余汉华,何怡刚. 基于果蝇算法的 Boltzmann 机谐波检测[J]. 微型机与应用,2012,31(18):66-68.

[6] 韩俊英,刘成忠. 自适应混沌混合果蝇优化算法[J]. 计算机应用,2013,33(5):1313-1316,1333.

[7] 韩俊英,刘成忠. 自适应调整参数的果蝇优化算法[J]. 计算机工程与应用,2014,50(7):50-55.

[8] 韩俊英,刘成忠,王联国. 动态双子群协同进化果蝇优化算法[J]. 模式识别与人工智能,2013,26(11):1057-1067.

[9] 刘成忠,韩俊英. 基于细菌迁徙的自适应果蝇优化算法[J]. 计算机工程与科学,2014,36(4):690-696.

[10] 张前图,房立清,赵玉龙. 具有 Lévy 飞行特征的双子群果蝇优化算法[J]. 计算机应用,2015,35(5):1348-1352.

[11] 胡 洁. 细菌觅食优化算法的改进及应用研究[D]. 武汉:武汉理工大学,2012.

[12] 韩俊英,刘成忠. 应用反向学习策略的果蝇优化算法[J]. 计算机应用与软件,2013,4:157-160.

[13] 任圆圆,刘培玉,薛素芝. 一种新的自适应动态文化粒子群优化算法[J]. 计算机应用研究,2013,30(11):3240-3243.

[14] 郑洪清,周永权. 一种自适应步长布谷鸟搜索算法[J]. 计算机工程与应用,2013,49(10):68-71.

Research of the Self-adaptive Step Fruit Fly Optimization Algorithm

DUAN Yanming [1], XIAO Huihui[1,2]

(1. College of Computer and Information Engineering, Hechi University, Yizhou 546300, China;
2. School of Information and Technology, Jiangxi University of Finance and Economics, Nanchang 330013, China)

Abstract: According to the problem that fruit fly optimization algorithm has low convergence accuracy, slow convergence velocity and easily falling into local optimization, we present a self-adaptive step fruit fly optimization algorithm (ASFOA). ASFOA can adjust adaptively the moving step according to the optimal flavor concentration values and the number of iterations during the evolution. The large step of ASFOA in the initial state ensure that the solution cannot be trapped into local optimum. While the small step of ASFOA in the later stage improves the convergence accuracy and computational efficiency. The simulation results of 6 standard benchmark functions show that the ASFOA algorithm has the advantages of better global searching ability, the improved algorithm is much better than basic FOA, FOAAM and ACFOA in the respects of convergence precision convergence speed

Keywords: adaptive; fruit fly optimization algorithm; convergence speed; taste concentration