·人工智能及识别技术· 文章编号：1000—3428(2013)05—0218—04 文献标识码：A 中图分类号：TP301.6

基于混沌映射的混合果蝇优化算法

程 慧，刘成忠
(甘肃农业大学工学院，兰州 730070)

摘 要：在果蝇算法的优化过程中，收敛精度会因为初值选取适当与否呈现不稳定状态。针对该问题，提出一种新的混合果蝇算法，该算法融入 Logistic 映射进行全局搜索得到最优参数值，再以该值为中心在其周围产生微小波动以获取初值进行二次寻优，改进果蝇算法中的初值选取方法。将该混合果蝇算法在函数优化中与原果蝇算法、粒子群算法等进行仿真对比，结果表明其在收敛精度方面具有明显优势。

关键词：果蝇优化算法；Logistic 映射；最优参数值；函数优化；收敛精度

Mixed Fruit Fly Optimization Algorithm Based on Chaotic Mapping

CHENG Hui, LIU Cheng-zhong
(College of Engineering, Gansu Agricultural University, Lanzhou 730070, China)

【Abstract】In the optimization process of the fruit fly algorithm, the convergence accuracy is unstable because of the initial value choosing suitable or not. In order to solve this problem, this paper proposes a new mixed fruit fly algorithm. The Logistic mapping is integrated into the fruit fly algorithm to do global search for the optimal parameter. It uses the value as the center to do tiny fluctuations to obtain initial value of quadratic optimization, and improve the initial value selection method of fruit fly algorithm. In the function optimization simulation process, compared with the original fruit fly algorithm and Particle Swarm Optimization(PSO) algorithm etc., the convergence accuracy of this mixed fruit fly algorithm has obvious advantages.

【Key words】Fruit Fly Optimization Algorithm(FOA); Logistic mapping; best parameter; function optimization; convergence precision
DOI: 10.3969/j.issn.1000-3428.2013.05.048

1 概述

果蝇优化算法(Fruit Fly Optimization Algorithm, FOA)是在 2011 年从果蝇的觅食行为中得到启发，而提出的一种寻求全局优化的新方法[1]，是一种新的群体智能算法，迄今为止，果蝇最佳化演算法属于一个全新的领域。FOA 可广泛应用于求解数学函数极值、微调 Z-SCORE 模型系数与混合类神经网络等各种领域[2]。由于 FOA 是一种刚被提出的全新算法，处于发展初期，并未被熟知，国内外学者对它的优缺点分析与改进方法并未做相关研究。该算法具有简单、收敛速度快且收敛精度高等优点，但是也存在一些问题，在算法的整个优化过程中收敛精度会因为初值的选取适当与否呈现不稳定状态，当初值选取较为适当时，收敛精度较高，若初值选取不适当，易陷入局部最优，且精度较低。混沌搜索具有遍历性、随机性、规律性等特点[3]，能在一定范围内按其自身的“规律”不重复地遍历所有状态[4]，在搜索过程中，可以避免陷入局部极小点[5]。但当搜索空间大时其效果却不能令人满意[6]。为了克服 FOA 收敛精度不稳定的问题，本文提出一种基于混沌映射的混合果蝇优化算法。利用 FOA 运算简单、收敛速度快、初值选取适当时收敛精度较高和混沌算法遍历性的特点，在运用融入混沌映射的果蝇算法进行全局搜索得到最优参数值的基础上，通过在最优参数值周围产生微小波动获取初值，达到二次寻优的目的。

2 基本概念

2.1 基于 Logistic 映射的混沌技术

混沌是自然界非线性系统中一种较为普遍的现象，其变化过程看似是混乱的，实际上其内在具有规律性。混沌变量是一个在[0,1]区间波动的变量，混沌现象具有随机性、

基金项目：国家自然科学基金资助项目(61063028)；甘肃省科技支撑计划基金资助项目(1011NKCA058)；甘肃省教育厅科研基金资助项目(0902-04)
作者简介：程 慧(1987－)，女，硕士研究生，主研方向：人工智能；刘成忠(通讯作者)，副教授、博士
收稿日期：2012-06-15 **修回日期：**2012-08-07 **E-mail：**liucz@gsau.edu.cn

遍历性和规律性以及对初值具有敏感性的特点[7]。混沌优化的基本思想是：将优化变量通过混沌映射规则映射到混沌变量空间的取值区间内，利用混沌变量的遍历性和规律性寻优搜索，将获得的优化解线性转化到优化空间[8]。通常所采用的 Logistic 映射是一个源于人口统计的动力学系统，其系统方程为[9]：

$$x(n+1)=ux(n)(1-x(n)) \tag{1}$$

其中，n 为迭代次数；$x(n)\in[0,1]$；u 是控制参数，当 $u=4$ 时，系统处于混沌状态。

混沌变量 Cx_i 的一种变换算式[10]如下：

$$Cx(n+1)_i=4Cx(n)_i\ (1-Cx(n)_i)\quad i=1,2,\cdots,N \tag{2}$$

其中，$Cx(n)_i$ 为映射的第 i 个混沌变量 Cx_i 在第 n 步混沌变换后的值，当 $Cx_i\in[0,1]$，且 $Cx_i\neq\{0.25,0.50,0.75\}$ 时，将产生混沌现象，式(2)的优化变量 $x_i\in[a_i,b_i]$，可由式(3)、式(4)与混沌变量 $Cx_i\in[0,1]$ 进行往返映射[10]。

$$Cx_i=(x_i-a_i)/(b_i-a_i) \tag{3}$$

$$x'_i=a_i+Cx_i(b_i-a_i) \tag{4}$$

其中，x'_i 为经混沌映射后的第 i 个混沌变量转化为常规优化变量而获得的值。由于在 Logistic 映射空间[0,1]内有 3 个断点，因此在进行混沌映射优化时应跳过 0.25、0.50、0.75 这 3 个断点。

2.2　果蝇优化算法

FOA 是一种从果蝇的觅食行为中得到启发的现代启发式算法[2]，该算法比较简单且容易实现，不需要调整过多的参数。果蝇依靠本身在嗅觉与视觉上的特性，其嗅觉器官能很好地搜集漂浮在空气中的各种气味，甚至能嗅到几十公里以外的食物源。使用敏锐的视觉发现食物与同伴聚集的位置，并往该方向飞去。依据果蝇搜索食物特性，将果蝇优化算法归纳为如下步骤：

(1)给定群体规模 *sizepop*，最大迭代数 max *gen*，随机初始果蝇群体位置用矢量 $\boldsymbol{x}\_1$、$\boldsymbol{y}\_1$ 表示。

(2)据下式计算果蝇个体利用嗅觉搜寻食物之随机方向与距离[2]，其中，*Value* 为搜索距离；矢量 $\boldsymbol{x}_i$ 为下一位置的横坐标；矢量 $\boldsymbol{y}_i$ 为下一位置的纵坐标：

$$\boldsymbol{x}_i=\boldsymbol{x}\_1+Value\times rand() \tag{5}$$

$$\boldsymbol{y}_i=\boldsymbol{y}\_1+Value\times rand() \tag{6}$$

(3)根据式(7)先估计与原点的距离矢量 $\boldsymbol{d}_i$，再利用式(8)计算味道浓度判定值矢量 $\boldsymbol{s}_i$[2]：

$$\boldsymbol{d}_i=\sqrt{\boldsymbol{x}_i^2+\boldsymbol{y}_i^2} \tag{7}$$

$$\boldsymbol{s}_i=1/\boldsymbol{d}_i \tag{8}$$

(4)将味道浓度判定值 $\boldsymbol{s}_i$ 代入味道浓度判定函数，用来求出果蝇个体位置的味道浓度，用矢量 $\boldsymbol{Smell}_i$[2]表示。

$$\boldsymbol{Smell}_i=Function(\boldsymbol{s}_i)$$

(5)找出该果蝇群体中最好的味道浓度值及最佳位置，分别用矢量 $\boldsymbol{Smell}_g$、$\boldsymbol{x}_g$、$\boldsymbol{y}_g$ 表示(求极小值及其所对应位置)。

(6)记录并保留最佳味道浓度值。变量 *Smellbest* = $\boldsymbol{Smell}_g$，初始位置为 $\boldsymbol{x}\_1=\boldsymbol{x}_g$，$\boldsymbol{y}\_1=\boldsymbol{y}_g$，这时果蝇群体利用视觉向该位置飞去。

(7)进入迭代寻优，重复执行步骤(2)~步骤(5)，并判断味道浓度是否优于前一迭代味道浓度，若是则执行步骤(6)。

3　改进的果蝇混合优化算法

选取最佳的参数值组合，使得训练模型与参考模型具有最小的均方误差。因为 FOA 不具有遍历性，而混沌算法具有遍历性，所以在一次迭代中，首先采用混沌映射搜索最优解所对应的参数值，即最优参数值，然后采用在一次迭代中获得的最优参数值，使其周围产生微小波动来获取二次迭代的初值，从而提高整个算法的求解精度，达到提高 FOA 优化搜索性能的目的。在 FOA 每次随机产生在[0,1]的值时，将[0,1]的值即优化变量通过 Logistic 映射到[0,1]混沌区间内，经过式(2)获得混沌序列，经式(4)变换后返回到优化解的空间内，利用混沌映射的混沌遍历性产生初次迭代的最优参数值，以获得的最优参数为中心，在其周围产生微小波动获取初值搜索最优值。改进的果蝇混合优化算法的主要步骤为：

(1)给定群体规模 *sizepop*，最大迭代数 max *gen*，随机产生 *sizepop* 个在 $[0,1]$ 的粒子的初始位置，用行向量 $\boldsymbol{z}_i$ 表示，(该 $\boldsymbol{z}_i$ 在实验中选取 10 维向量，前 5 位相当于 FOA 中 $\boldsymbol{x}_i$ 的值，后 5 位相当于 $\boldsymbol{y}_i$ 的值。一般根据优化参数个数来定义 $\boldsymbol{z}_i$ 的维数)。

(2)将各 $\boldsymbol{z}_i$ 的各分量通过式(3)的变换，映射为混沌变量 $Cz(n)_i$，且 $Cz(n)_i\in[0,1]$。

(3)将混沌变量 $Cz(n)_i$ 的各分量经式(2)做混沌操作。

(4)将每个分量通过式(4)变换，映射为 $[a_1,b_1]$ 间的普通变量 $\boldsymbol{z}'_i$，并计算适应度 $f(\boldsymbol{z}'_i)$；选取种群中最小的 $f(\boldsymbol{z}'_i)$；记录选取的最小 $f(\boldsymbol{z}'_i)$，矢量 $\boldsymbol{fit}(gen)=f(\boldsymbol{z}'_i)$。

(5)进入初次迭代寻优，重复执行步骤(3)~步骤(4)。

(6)判断迭代次数是否等于最大迭代终止次数，若是，则选取最小 $\boldsymbol{fit}$，使 *Smellbest* = min($\boldsymbol{fit}$(*gen*))，此处记录最小 $f(\boldsymbol{z}'_i)$ 所对应的味道浓度矢量 $\boldsymbol{S}_g$ (并非是 FOA 中最好的初始位置 $\boldsymbol{z}_i$，而是参数值)。

(7)要在初次迭代的最优参数值周围产生微小波动，即要获得的初值在初次迭代的最优参数的小领域内，若参数大于 0，则 $B\in[0,1]$，经反复验证，证实结果较好，这里选取 $B=0.25$，在选取的味道浓度 $\boldsymbol{S}_g$ 周围利用 $\boldsymbol{S}_i=\boldsymbol{S}_g+2Brand()-B$ 产生微小波动种群，并利用 $\boldsymbol{S}_i$ 计算 $f(\boldsymbol{S}_i)$，选取种群中最小的 $f(\boldsymbol{S}_i)$，并且有 *Bestsmell* = min($f(\boldsymbol{S}_i)$)，如果最小的 *Bestsmell* < *Smellbest*，那么

$Smellbest = Bestsmell$，记录 $\boldsymbol{S}_g = \boldsymbol{S}_i$。

(8)进入二次迭代寻优，更新粒子味道浓度值 $\boldsymbol{S}_i$，重复执行步骤(7)。

(9)判断迭代次数是否等于最大迭代终止次数，若是，输出 $Smellbest$，$P = \boldsymbol{S}_g$。

4 实验结果与分析

为了验证本文算法，将该算法应用在函数求解中。采用 benchmark 测试算法的权威优化问题作为实例，并与常规粒子群优化(Particle Swarm Optimization, PSO)算法、文献[9]算法、文献[11]算法、文献[12]算法对比。该优化实例有 5 个优化变量(x_1,x_2,x_3,x_4,x_5)，非线性不等式以及边界约束条件具体形式为[11]:

$$\min f(\boldsymbol{X}) = 5.357\,854x_3^2 + 0.835\,689\,1x_1x_5 + 37.293\,29x_1 - 40\,729.141$$

$$y_1(\boldsymbol{X}) = 85.334\,407 + 0.005\,685\,8x_2x_5 + 0.000\,26x_1x_4 - 0.002\,205\,3x_3x_5$$

$$y_2(\boldsymbol{X}) = 80.512\,49 + 0.007\,131\,7x_2x_5 + 0.002\,995\,5x_1x_2 + 0.002\,181\,3x_3^2$$

$$y_3(\boldsymbol{X}) = 9.300\,961 + 0.004\,702\,6x_3x_5 + 0.001\,254\,7x_1x_3 + 0.001\,908\,5x_3x_4$$

$$0 \leqslant y_1(\boldsymbol{X}) \leqslant 92$$

$$90 \leqslant y_2(\boldsymbol{X}) \leqslant 110$$

$$19 \leqslant y_3(\boldsymbol{X}) \leqslant 25$$

$$78 \leqslant x_1 \leqslant 102$$

$$33 \leqslant x_2 \leqslant 102$$

$$27 \leqslant x_3 \leqslant 45$$

$$27 \leqslant x_4 \leqslant 45$$

$$27 \leqslant x_5 \leqslant 45$$

其中，向量 $\boldsymbol{X} = (x_1, x_2, \cdots, x_5)^{\mathrm{T}}$。

基于上述优化实例，在 Matlab R2010b 软件环境下，采用具有 1.73 GHz 的主处理器并拥有 1 GB 内存的 PC 机，每种优化方法独立运行 100 次，并统计其运行结果。在实验中，粒子数 $sizepop$ =30；最大迭代终止次数 $\max gen$ = 1 000，第 3 节算法步骤(6)、步骤(9)的最大迭代终止次数可以依情况设定，此处最大迭代终止次数都为 $\max gen$。

表 1 给出了采用传统 FOA 算法、PSO 算法、本文算法、以及文献[9]中 2 种算法 100 次独立运行的结果。

表 1 5 种算法的优化结果对比

算法	最差解	最优解	平均 $f(\boldsymbol{X})$	迭代平均运行时间/s	平均有效迭代次数
传统 FOA 算法	−28 758	−31 195	−30 286	9.0	35
PSO 算法	−27 802	−30 725	−28 782	19.0	845
文献[9]算法 1	−28 904	−31 047	−29 491	31.0	723
文献[9]算法 2	−29 621	−31 109	−30 194	10.0	969
本文算法	−31 537	−31 544	−31 540	8.0	472

由表 1 可知，本文算法收敛精度明显优于其他方法。上述函数目的是求解最小值，从平均 $f(\boldsymbol{X})$看，本文算法所求平均最优解为−31 540，不仅优于 FOA 算法，而且效果比文献[9]的 2 个算法也好。从最优解和最差解上看，本文算法所求的 $f(\boldsymbol{X})$分别为−31 544、−31 537，与其他算法相比也是最好的，且本文算法最差解−31 537 比其他算法的最优解都小，由此看出本文算法的精确度最好的，从迭代平均运行时间以及平均有效迭代次数 2 项指标来考查，本文算法平均有效迭代次数 472，与 FOA 算法和其他文献方法比较，虽然高于 FOA，但与其他算法比较是较低的，且算法平均运行时间 8 s，其值明显小于其他优化方法的，总体来说，本文算法结合了 FOA 收敛速度较快和混沌算法遍历性的优点，克服 FOA 收敛精度较低且不稳定的缺点。较之 FOA、PSO、文献[9]改进算法，收敛精度和稳定性较好。

表 2 给出了采用传统 FOA 算法、PSO 算法、本文算法、文献[9,11-12]算法在进行 100 次独立运行中获得最优结果时，各变量 X_1、X_2、X_3、X_4、X_5 的值以及各约束值 y_1、y_2、y 的值。

表 2 6 种算法获得最优结果时的变量值与约束值对比

优化变量	传统 FOA 算法	PSO 算法	文献[12]	文献[11]	文献[9]	本文算法
x_1	82.775 3	79.942 5	78.000 0	78.049 5	78.990 2	78.004 2
x_2	40.115 9	34.018 0	33.000 0	33.007 0	34.275 0	34.844 0
x_3	33.020 1	27.909 2	29.995 0	27.081 0	28.093 4	27.003 3
x_4	35.926 0	44.103 0	45.000 0	45.000 0	45.000 0	44.986 4
x_5	27.943 7	43.591 0	36.776 0	44.940 0	37.981 9	37.309 9
y_1	91.026 1	91.999 5	90.714 7	91.997 6	91.307 4	91.416 7
y_2	99.258 5	100.933 2	98.840 5	100.407 8	99.628 3	99.516 1
y_3	18.258 7	20.170 6	19.999 9	20.001 9	19.515 9	19.000 0
$f(\boldsymbol{X})$	−31 195	−30 725	−30 665.609 0	−31 020	−31 109	−31 544

由表 2 可知，从各算法的最优解上看，采用本文算法获得该优化问题的最优解−31 544，其相应参数变量的值分别为：x_1=78.004 2，x_2=34.844 0，x_3=27.003 3，x_4=44.986 4，x_5=37.309 9，其最优解−31 544 较之其他算法的−31 195、−30 725、−30 665.609 0、−31 020、−31 109，−31 544 最小，故该算法获得解的精度比其他算法高。

图 1 将 FOA 算法与本文算法在 100 次独立实验后，最优结果进行对比，从图 1 中看出，FOA 收敛速度较快，在第 15 次就取得最优值，而本文算法在 320 次迭代时取得最优值，但是本文算法获得的最小值为−315 44。FOA 算法的最小值为−31 195，故本文算法收敛精度较高，由此看出，FOA 算法极易陷入局部最优，取得的最优值明显大于本文算法，收敛精度较低。所以，从整体看，本文算法收敛精度较高且较稳定，相对于其他算法，其收敛速度较快。

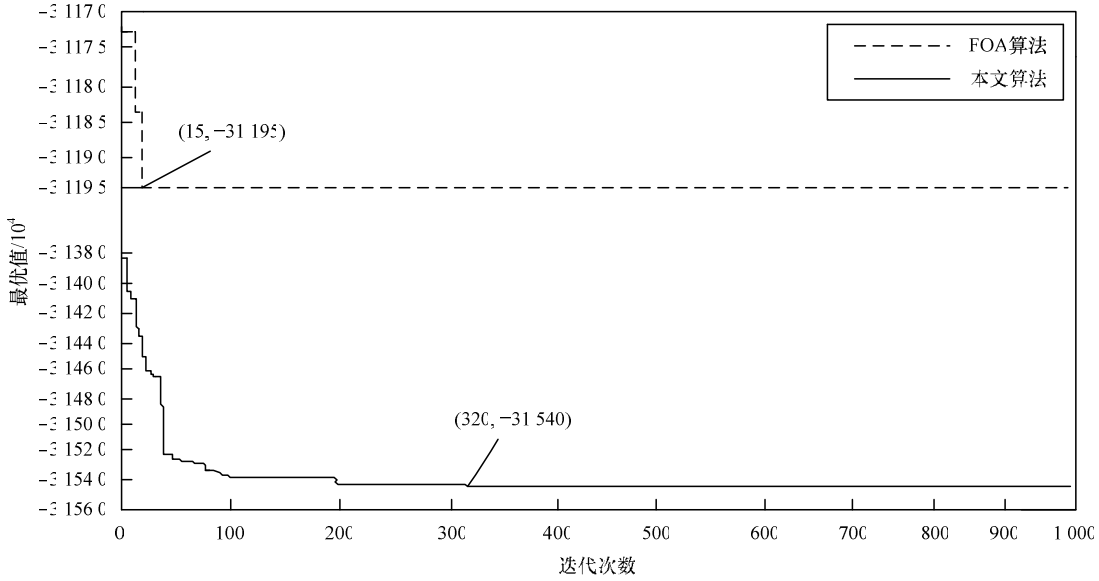

图 1　2 种算法最优值对比

5　结束语

本文提出一种基于混沌映射的混合果蝇优化算法。在初次迭代中融入 Logistic 映射，并在初次迭代的最优值周围通过产生微小波动获取二次迭代初值，实验结果表明该算法具有较好的收敛效果。由于本文算法运行时间相对较长，且需要跳过 3 个断点，复杂度相对较高，因此下一步的工作是加快训练速度，用 Lozi's 映射来代替 Logistic 映射，以降低复杂度。

参考文献

[1] 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, 2011, 29(4): 1-5.

[2] 潘文超. 果蝇最佳化演算法[EB/OL]. (2010-11-21). http://www.ilovematlab.cn/forum.php?mod=viewthread&tid=143951.

[3] 李　兵, 蒋慰孙. 混沌优化方法及其应用[J]. 控制理论与应用, 1997, 14(4): 613-615.

[4] 柳　贺, 黄　猛, 黄　道. 基于混沌搜索的优化方法的研究进展[J]. 南京理工大学学报: 自然科学版, 2005, 29(S1): 124-128.

[5] Liu Shengsong, Hou Zhijian, Wang Min. A Hybrid Algorithm for Optimal Power Flow Using the Chaos Optimization and the Linear Interior Point Algorithm[C]// Proc. of International Conference on Power System Technology. [S. l.]: IEEE Press, 2002.

[6] 张　彤, 王宏伟, 王子才. 变尺度混沌优化方法及其应用[J]. 控制与决策, 1999, 14(3): 285-287.

[7] 吕晓明, 黄考利, 连光耀. 基于混沌粒子群优化的系统级故障诊断策略优化[J]. 系统工程与电子技术, 2010, 32(1): 217-220.

[8] 杨俊杰, 周建中, 喻　菁, 等. 混合混沌优化方法及其在非线性规划问题中的应用[J]. 计算机应用, 2004, 24(10): 119-120.

[9] 刘道华, 原思聪, 兰　洋, 等. 混沌映射的粒子群优化方法[J]. 西安电子科技大学学报, 2010, 37(4): 765-768.

[10] 莫愿斌, 陈德钊, 胡上序. 混沌粒子群算法及其在生化过程动态优化中的应用[J]. 化工学报, 2006, 57(9): 2123-2127.

[11] Carlos A, Coello C. Use of a Self-adaptive Penalty Approach for Engineering Optimization Problems[J]. Computers in Industry, 2000, 41(2): 113-127.

[12] Homaifar A, Charlene X Q, Steven H L. Constrained Optimization via Genetic Algorithms[J]. Simulation, 1994, 62(4): 242-253.

编辑　刘　冰