·人工智能及识别技术· 文章编号:1000-3428(2017)02-0210-05 文献标志码:A 中图分类号:TP391

基于近郊区和远郊区的果蝇优化新算法

王友卫[1],朱建明[1],凤丽洲[2],李 洋[1]

(1. 中央财经大学 信息学院,北京 100081; 2. 吉林大学 计算机科学与技术学院,长春 130012)

摘 要: 在传统果蝇优化算法中,果蝇的新位置常被限定在特定区域内,因此,寻优结果对搜索半径依赖性强,导致算法极易陷入局部最优。为此,提出一种改进的果蝇优化算法。将果蝇在每个维度上的搜索范围分为 2 个部分,给出近郊区和远郊区的概念,引入局部最优导向因子,通过动态调整该因子协调果蝇在不同区域的搜索强度,通过随机选择果蝇位置向量中特定维度实现果蝇位置更新。仿真实验结果表明,与传统自适应混沌果蝇优化算法相比,该算法能有效避免搜寻半径的影响,且在收敛精度、收敛速度等方面具有明显优势。

关键词: 果蝇优化算法;局部最优;搜寻半径;收敛精度;收敛速度

中文引用格式: 王友卫,朱建明,凤丽洲,等. 基于近郊区和远郊区的果蝇优化新算法[J]. 计算机工程,2017,43(2):210-214.

英文引用格式: Wang Youwei,Zhu Jianming,Feng Lizhou,et al. Novel Fruit Fly Optimization Algorithm Based on Near Suburb and Far Suburb[J]. Computer Engineering,2017,43(2):210-214.

Novel Fruit Fly Optimization Algorithm Based on Near Suburb and Far Suburb

WANG Youwei[1],ZHU Jianming[1],FENG Lizhou[2],LI Yang[1]

(1. School of Information,Central University of Finance and Economics,Beijing 100081,China;
2. College of Computer Science and Technology,Jilin University,Changchun 130012,China)

【Abstract】 In the traditional Fruit Fly Optimization Algorithm(FOA),the new positions of fruit flies are often limited to particular regions,thus the optimization results are highly dependent on the searching radiuses,and are easy to fall into local optimum. On this basis,an improved FOA algorithm is proposed. The search scope of the fruit fly in each dimension is divided into two parts,and the conceptions of near suburb and far suburb are introduced. A local optimum oriented factor is introduced,and the searching intensities of different scopes are coordinated by adjusting this factor. The fruit fly position is updated by randomly selecting a specific dimension of it. Simulation results show that,when compared with traditional adaptive chaos FOA methods,the proposed method can avoid the effect of searching radius effectively,and has evident advantages on convergence accuracy and convergence speed.

【Key words】 Fruit Fly Optimization Algorithm(FOA); local optimum; searching radius; convergence accuracy; convergence rate

DOI: 10.3969/j.issn.1000-3428.2017.02.035

0 概述

果蝇优化算法(Fruit Fly Optimization Algorithm,FOA)是一种模拟果蝇觅食行为的新的全局优化进化算法[1-2]。与传统参数寻优方法(粒子群寻优、遗传算法、免疫算法等)相比,FOA 算法不仅实现简单、耗时小,而且所需参数较少,能有效避免传统参数寻优算法中参数选取困难及对寻优结果影响较大的问题[3-4]。但是,传统 FOA 算法对果蝇搜索半径依赖性强,导致算法极易陷入局部最优。

文献[5]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。但此方法针对离散函数寻优效果不明显,并且二次寻优过程增加了算法的计算开销。文献[6]提出了一种自适应果蝇优化算法,该算法在收敛后使用混沌映射产生果蝇新位置,但缺点在于无法寻优负值参数。文献[7]对传统 FOA 算法做出改进,引入逃离参数 δ 实现对负值参数的寻优。

基金项目: 信息保障技术重点实验室开放基金(KJ-14-008)。

作者简介: 王友卫(1987—),男,讲师、博士,主研方向为机器学习、信息隐藏、数据挖掘;朱建明,教授、博士、博士生导师;凤丽洲,讲师、博士;李 洋,副教授、博士。

收稿日期: 2015-11-16 **修回日期:** 2016-02-03 **E-mail:** ywwang15@126.com

但由于 δ 与果蝇位置存在非线性关系,因此 δ 的选择对于寻优结果影响较大。文献[8]将果蝇优化算法应用于连续查询攻击,通过采用种群的平均味道浓度来替换浓度最大味道浓度,有效避免了算法陷入局部最优的现象。文献[9]去掉了气味浓度判定值及距离 2 个参数,使用果蝇位置向量直接计算适应度函数,每次更新果蝇位置时只选择对应向量中某一维度进行更新。文献[10]提出了一种基于最优和最差个体协同学习的果蝇优化算法。算法通过在进化方程中添加向最差个体学习的改进策略,优化进化方程,增强算法跳出局部最优、寻找全局最优的能力。

为克服传统 FOA 算法对果蝇搜索半径依赖性强、算法极易陷入局部最优等问题,本文根据果蝇当前位置引入近郊区、远郊区、局部最优导向因子等相关概念,在此基础上,提出一种结合近郊区和远郊区的果蝇优化新算法。算法通过控制果蝇在不同区域的搜索强度协调算法的局部细化和全局搜索性能,在保证收敛精度的同时实现算法的全局寻优。

1　传统果蝇优化算法

果蝇在觅食过程中凭借敏锐的嗅觉和视觉发现食物最佳位置,PAN 据此于 2011 年提出了果蝇优化算法 FOA[1]。该算法主要执行步骤如下[1,11-12]:

步骤 1　初始化参数。包括种群中果蝇数量 N、最大迭代次数 T、搜索范围 $[x_{\min},x_{\max}]$、果蝇的初始位置 $(x_{\_axis},y_{\_axis})$ 等。

步骤 2　按照下面公式随机产生果蝇的新位置 (x_i,y_i):

$$x_i = x_{\_axis} + RandomValue$$
$$y_i = y_{\_axis} + RandomValue \tag{1}$$

其中,$RandomValue$ 为随机产生的搜索距离。

步骤 3　计算果蝇对应的气味浓度判定值 s_i,如下:

$$d_i = \sqrt{x_i^2 + y_i^2}, s_i = \frac{1}{d_i} \tag{2}$$

步骤 4　将每只果蝇对应的气味浓度判定值 s_i 带入适应度函数 Fit 中求出该果蝇的气味浓度 $Smell_i$:

$$Smell_i = Fit(s_i) \tag{3}$$

步骤 5　找出群体中气味浓度最大的果蝇,将该果蝇位置及对应的气味浓度值分别记为 (x_b,y_b),$Smell_b$。

步骤 6　将位置 (x_b,y_b) 作为下一次产生新果蝇的初始位置。

步骤 7　循环执行步骤 2 ~ 步骤 6 过程直到达到最大迭代次数,此时算法结束。

2　本文方法

给定当前搜索半径 r,搜索范围 $[x_{\min},x_{\max}]$,当前果蝇最优位置向量 $\boldsymbol{X}_b = \{x_{bj}\}$ $(0 \leqslant j < d, d$ 为向量维度),可按下面公式将果蝇位置向量中每个维度分为 2 个部分:

$$\Omega_1 = \{x_{ij} \mid x_{bj} - r \leqslant x_{ij} \leqslant x_{bj} + r\} \tag{4}$$
$$\Omega_2 = \Omega - \Omega_1 \tag{5}$$

其中,Ω 为针对每个维度的搜索范围。如图 1 所示,本文分别将 Ω_1,Ω_2 称为当前维度最优值 x_{bj} 的近郊区、远郊区。针对某寻优问题,假设当前维度的目标值在图 1 中 x_{gj} 处,即处于 x_{bj} 的远郊区。

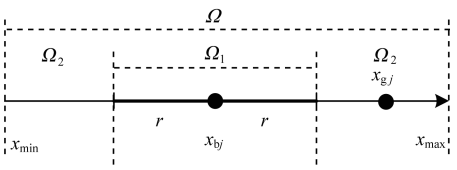

图 1　近郊区与远郊区示意图

由于文献[1,8-9]在每次迭代过程中只在当前最优值的近郊区内搜索,因此将导致算法无法找到目标位置 x_{gj},继而陷入局部最优。文献[6]在算法局部收敛后结合混沌映射产生果蝇新位置,虽然能提高算法跳出局部最优的可能性,但仍存在 2 个问题:1)混沌映射过程本身耗时较大;2)混沌映射所产生的果蝇新位置仍易落入当前最优值的近郊区域,因此,搜索过程带有很大的盲目性。

为此,本文对文献[9]进行改进,引入局部最优导向因子 p,结合当前最优值的近郊区和远郊区提出了一种新的果蝇位置更新过程,具体如下:

输入　当前搜索半径 r,搜索范围 $[x_{\min},x_{\max}]$,当前果蝇最优位置向量 $\boldsymbol{X}_b = \{x_{bj}\}$ $(0 \leqslant j < d, d$ 为向量维度),当前果蝇位置向量 $\boldsymbol{X}_i$,局部最优导向因子 p

输出　果蝇 $\boldsymbol{X}_i$ 在第 j 维度上的新位置 x_{ij}

begin

1. if Rand( ) ≤ p

$$x_{ij} \leftarrow x_{bj} + (-1)^{\lfloor 10 \times Rand() \rfloor} \times r \times Rand() \tag{6}$$

2. else if (($x_{bj} + r < x_{max}$) and ($x_{bj} - x_{min} \leqslant r$))

$$x_{ij} \leftarrow x_{bj} + r + (x_{max} - x_{bj} - r) \times Rand() \tag{7}$$

3. else if (($x_{bj} - x_{min} > r$) and ($x_{bj} + r < x_{max}$))

$$v \leftarrow x_{bj} + r + (x_{max} - x_{min} - 2 \times r) \times Rand() \tag{8}$$

3.1. if $v < x_{max}$

$$x_{ij} \leftarrow v \tag{9}$$

3.2. else

$$x_{ij} \leftarrow x_{min} + v - x_{max} \tag{10}$$

3.3. end if

4. else

$$x_{ij} \leftarrow x_{min} + (x_{bj} - r) \times Rand() \tag{11}$$

5. end if

end

其中,Rand( ) 函数随机生成一个区间 [0,1] 内的随机数,⌊ ⌋ 为向下取整函数。步骤 1 对应于当前最优值近郊区的寻优过程,步骤 2 ~ 步骤 5 则对应算法在当前最优值远郊区的寻优过程。可见,本文不仅结合搜索半径 r 生成果蝇新位置,而且结合 p 值选择远郊区生成果蝇新位置。因此,新位置生成过程不

再单纯依赖于当前最优值位置、搜寻半径等参数,而是通过在一定程度上使用远郊区生成果蝇新位置增强算法的全局寻优能力。进一步地,将参考文献[9]并按照下面公式更新 r,p:

$$r = r_{\max} \times \exp\left(\mathrm{lb}\left(\frac{r_{\min}}{r_{\max}} \right) \times \frac{t}{t_{\max}} \right) \tag{12}$$

$$p = p_{\max} \times \exp\left(\mathrm{lb}\left(\frac{p_{\min}}{p_{\max}} \right) \times \frac{t}{t_{\max}} \right) \tag{13}$$

其中,$p_{\min} = 0.001$;$p_{\max} = 1$。可见,为实现目标值的快速定位,p 值在迭代起始阶段较大,此时本文将参照文献[9]方法以较大概率在当前最优值的近郊区搜索;随着 p 值逐渐减小,本文逐渐提高了针对当前最优值远郊区的搜索能力,以避免算法陷入局部极值。综上,本文执行步骤如下所示。

输入 果蝇种群 X(数量为 N),最大迭代次数 T,当前迭代次数 t,向量维度 d,搜索半径 r 最大值 $r_{\max}$,搜索半径 r 最小值 $r_{\min}$,搜索范围[$x_{\min}$, $x_{\max}$],临时变量 i

输出 全局最优果蝇 $\boldsymbol{X}_b$

begin

1. i←0;
2. for (;$X_i \in X$;i + +)

//按照下面公式随机生成 X_i 的位置:

$$x_{ij} \leftarrow x_{\min} + (x_{\max} - x_{\min}) \times \mathrm{Rand}(), 0 \leqslant j < d \tag{14}$$

其中,x_{ij} 为果蝇 X_i 的第 j 维数值。

3. end for
4. i←0;
5. for (;$X_i \in X$;i + +)

计算所有果蝇的适应值 fit(X_i);

6. end for
7. 找到对应适应值最大的果蝇并将其记为 X_b;
8. t←0;
9. while(t < T)
10. i←0;
11. for(;$X_i(X_i \neq X_b) \in X$;i + +)

11.1. 随机选择 X_i 中某一维 x_{ij};

11.2. 按照果蝇位置更新过程更新 x_{ij} 的值;

11.3. 计算 X_i 适应值 fit(X_i);

12. end for
13. 求得对应适应值最大的果蝇 X_b;
14. 按照式(12)、式(13)更新 r,p;
15. end while
16. end

其中,步骤 1 ~ 步骤 7 为果蝇位置初始化及当前最优果蝇计算过程;步骤 8 ~ 步骤 15 为通过迭代方式搜索全局最优果蝇位置过程;步骤 11.2 为前面介绍的结合近郊区和远郊区的果蝇位置更新过程。

3 实验结果与分析

3.1 实验设置

为测试不同算法的性能,这里将分别从搜索半径的影响、时间复杂度、寻优精度、收敛速度 4 个方面将本文与文献[1,6,9]等方法进行比较。分别选取 3 个单峰基准函数(F_1 ~ F_3)、3 个多峰基准函数(F_4 ~ F_6)进行实验[13-15]。不同函数公式、变量取值范围、目标值及目标值位置如表 1 所示。公平起见,设置不同算法果蝇数量 N = 50,最大迭代次数 T = 2 000。

表 1 不同函数公式、变量取值范围、目标值及目标值位置

函数	公式	变量取值范围	目标值	目标值位置
F_1:Axis	$f(x) = \sum_{i=2}^{d} i x_i^2$	[-5.12,5.12]	0	(0,0,…,0)
F_2:Rosenbrock	$f(x) = \sum_{i=2}^{d} (100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-30,30]	0	(1,1,…,1)
F_3:Schwefel	$f(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	[-10,10]	0	(0,0,…,0)
F_4:Rastrigin	$f(x) = \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-100,100]	0	(0,0,…,0)
F_5:Schaffer	$f(x) = \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2} + 0.5$	[-100,100]	0	(0,0,…,0)
F_6:Generalized	$f(x) = \frac{\pi}{d} \left\{ \begin{array}{l} 10\sin^2(\pi y_1) \\ + \sum_{i=1}^{d-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] \\ + (y_d - 1)^2 \end{array} \right.$ $+ \sum_{i=1}^{d} \mu(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $\mu(x_i, a, k, m) = \left\{ \begin{array}{l} k(x_i - a)m, x_i > a \\ 0, -a \leqslant x_i \leqslant a \\ k(-x_i - a)m, x_i < -a \end{array} \right.$	[-50,50]	0	(-1,-1,…,-1)

3.2　搜索半径 r 对寻优结果的影响

为了测试搜索半径对算法寻优结果的影响,令:

$$r_{\max}=\alpha\times(x_{\max}-x_{\min}),r_{\min}=0.000\ 1 \tag{15}$$

其中,α 为搜索半径缩放因子,$\alpha\in(0,1)$。进一步地,设置维度 $d=100$,计算 α 取值不同情况下不同算法针对不同测试函数所得全局最优值均值,结果如表 2 所示。

表 2　搜索半径 r 对不同算法的寻优结果影响

a	文献[1]算法	文献[6]算法	文献[9]算法	本文算法
2^{-1}	1.32E-4	2.31E-5	1.32E-6	1.39E-6
2^{-2}	5.83E-5	3.59E-4	3.57E-5	8.17E-5
2^{-3}	7.66E-3	6.77E-3	5.66E-4	2.28E-5
2^{-4}	4.39E4	8.12E-4	2.89E-3	3.55E-6
2^{-5}	2.53E-3	8.66E-3	1.63	7.32E-5
2^{-6}	3.22E-3	5.19E-4	3.95	6.95E-5

由表 2 可知,文献[1,6]所得结果普遍偏低且波动较大;文献[9]表现随 α 值变小而逐渐变差;本文结果明显高于其他算法且相对平稳。可见,本文使用参数 p 实现对于当前最优值近郊区和远郊区的同步搜索,有效避免了搜索半径对寻优结果的影响,提高了算法的全局寻优能力。

3.3　时间复杂度分析

时间复杂度是衡量寻优算法性能优劣的重要指标之一。FOA 算法主要耗时过程包括生成果蝇初始位置、更新果蝇位置、计算果蝇适应值、更新参数4 个阶段。其中,计算果蝇适应值与具体适应值函数有关,故这里不做考虑。为此,将另外 3 个阶段时间复杂度分别记为 T_1,T_2,T_3,在此基础上计算不同寻优算法对应时间复杂度,结果如表 3 所示(其中,N 为果蝇数量;t 为 Logistic 混沌映射进入混沌状态所需的迭代次数;d 为向量空间维度)。对比发现,本文算法计算耗时与文献[9]算法一致。由于本文在每次果蝇位置更新中只选择特定维度,并且更新过程未使用混沌映射进行全局寻优,因此对应时间复杂度明显小于其他算法。

表 3　不同算法对应的时间复杂度比较

阶段时间复杂度	文献[1]算法	文献[5]算法	文献[6]算法	文献[9]算法	文献[10]算法	本文算法
T_1	$O(d\times N)$	$O(d\times N)$	$O(d\times N)$	$O(d\times N)$	$O(d\times N)$	$O(d\times N)$
T_2	$O(d\times N\times T)$	$O(d\times N\times T\times t)$	$O(d\times N\times T+d\times N\times T\times t)$	$O(N\times T)$	$O(d\times N\times T)$	$O(N\times T)$
T_3	–	–	–	$O(T)$	$O(T)$	$O(T)$
总时间复杂度	$O(d\times N\times T)$	$O(d\times N\times T\times t)$	$O(d\times N\times T\times t)$	$O((d+T)\times N)$	$O(d\times N\times T)$	$O((d+T)\times N)$

3.4　收敛精度比较

为降低实验误差,令维度 d 分别取 10,30,50,70,90,并分别针对每个 d 值进行 10 次实验,进一步按照下面公式计算 10 次实验所得函数最优值均值 E 和平均方差 D:

$$E=\frac{1}{50}\sum_{i,n}f_{i,n}$$

$$D=\frac{1}{50}\sum_{i,n}(f_{i,n}-E) \tag{16}$$

其中,i 表示针对每个 d 值进行的实验次数,i 属于[1,10];$f_{i,d}$ 表示当维度为 d 时第 i 次实验所得函数最优值。表 4 比较了不同方法针对不同测试函数的寻优精度,其中,符号 ± 前、后的数值分别为按照式(15)计算的函数最优值均值 E、平均方差 D。由表 4 知,由于无法针对负值参数进行寻优,因此在处理函数 F_6 时,文献[1,6,10]表现最差;在处理单峰函数 $F_1\sim F_3$ 时,本文所得结果较其他算法优势并不明显,这主要是因为单峰函数在定义域内只存在一个极值,寻优结果受搜寻半径影响较小;而在处理函数 $F_4\sim F_5$ 时,本文相对于其他算法优势明显,说明通过近郊远郊策略能获得较高的搜寻精度和稳定性。

表 4　不同算法的收敛精度比较

函数	文献[1]算法	文献[6]算法	文献[9]算法	文献[10]算法	本文算法
F_1	1.33E-6 ± 2.12E-6	6.46E-5 ± 4.22E-7	3.24E-5 ± 2.54E-7	5.32E-6 ± 5.17E-6	6.22E-6 ± 5.76E-7
F_2	2.53E-6 ± 4.57E-4	2.57E-7 ± 3.36E-6	5.36E-7 ± 3.14E-7	6.52E-5 ± 2.52E-4	2.57E-6 ± 3.36E-6
F_3	2.49E-7 ± 6.64E-7	4.35E-7 ± 4.26E-7	3.64E-7 ± 4.62E-6	5.44E-5 ± 7.67E-7	6.29E-7 ± 4.82E-7
F_4	5.38E-5 ± 5.12E-6	3.36E-6 ± 1.25E-4	1.43E-5 ± 5.62E-7	8.58E-5 ± 6.13E-6	7.13E-8 ± 5.35E-4
F_5	6.52E-5 ± 3.67E-9	6.65E-5 ± 5.32E-6	1.25E-6 ± 4.55E-6	8.32E-5 ± 5.93E-5	7.38E-8 ± 6.32E-6
F_6	3.41E-1 ± 2.12E-7	7.11E-2 ± 4.16E-6	7.35E-4 ± 6.12E-6	5.29E-2 ± 5.65E-4	9.43E-6 ± 5.65E-7

3.5 收敛速度比较

为了测试不同算法的收敛速度,这里针对每种寻优算法在每个测试函数上实验 100 次,统计每种算法达到收敛状态时的平均迭代次数 *NC*,结果如表 5 所示。为了验证算法是否达到目标最优值,表 5 还给出了 100 次实验中每种算法达到目标值的次数 *H*。由表 5 知,文献[1]收敛速度最快,但对应 *H* 值普遍偏小,说明该算法易过早陷入局部极值。本文对应 *NC* 值与文献[9]相近,且普遍小于文献[6,10],这说明本文的近郊区、远郊区搜索策略能有效加快算法的收敛速度。观察 *H* 值发现,本文对应结果普遍高于其他算法,进一步说明本文具有更高的全局寻优能力。

表 5 不同算法平均迭代次数及达到目标值次数的比较

函数	文献[1]算法		文献[6]算法		文献[9]算法		文献[10]算法		本文算法	
	NC	*H*	*NC*	*H*	*NC*	*H*	*NC*	*H*	*NC*	*H*
F_1	1 301	97	1 389	98	1 386	99	1 515	97	1 434	99
F_2	1 414	96	1 626	98	1 409	99	1 471	99	1 417	99
F_3	1 289	99	1 518	100	1 509	100	1 330	97	1 499	100
F_4	1 305	82	1 574	94	1 489	92	1 558	89	1 481	94
F_5	1 530	80	1 709	96	1 634	94	1 489	89	1 650	95
F_6	1 475	88	1 408	95	1 597	91	1 715	90	1 501	97

4 结束语

本文提出一种基于近郊区和远郊区的果蝇寻优算法。为降低算法对于搜索半径的依赖性,根据当前最优果蝇位置,给出近郊区和远郊区的概念。为提高果蝇算法的全局寻优能力,引入局部最优导向因子,并使用该因子控制果蝇在不同区域的搜索情况。基于多个测试函数的对比实验表明,该算法对搜索半径依赖性小,在算法收敛精度、收敛速度等方面相对于传统果蝇优化算法具有一定优势。由于本文测试的函数数量有限,因此下一步的工作重点是将本文算法应用于更加复杂的实际优化问题中。

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(社会科学版),2011,29(4):1-5.

[2] Pan W T. A New Fruit Fly Optimization Algorithm: Taking the Financial Distress Model as an Example[J]. Knowledge-based Systems,2012,26(2):69-74.

[3] Niu Jinwei, Zhong Weimin, Liang Yi, et al. Fruit Fly Optimization Algorithm Based on Differential Evolution and Its Application on Gasification Process Operation Optimization [J]. Knowledge-based Systems, 2015, 88(C):253-263.

[4] 吴小文,李 擎.果蝇算法和 5 种群智能算法的寻优性能研究[J].火力与指挥控制,2013,38(4):17-20,25.

[5] 程 慧,刘成忠.基于混沌映射的混合果蝇优化算法[J].计算机工程,2013,39(5):218-221.

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

[7] Li Chunquan, Xu Shaoping, Li Wen, et al. A Novel Modified Fly Optimization Algorithm for Designing the Self-tuning Proportional Integral Derivative Controller[J]. Journal of Convergence Information Technology,2012,7(16):69-77.

[8] 杨 琼,俞立峰,陈小小.一种基于果蝇优化方法的连续查询攻击算法[J].四川大学学报(自然科学版),2014,51(4):725-730.

[9] Pan Quanke, Sang Hongyan, Duan Junhua, et al. An Improved Fruit Fly Optimization Algorithm for Continuous Function Optimization Problems [J]. Knowledge-based Systems,2014,62(5):69-83.

[10] 韩俊英,刘成忠.反向认知的高效果蝇优化算法[J].计算机工程,2013,39(11):223-225,239.

[11] Zheng Xiaolong, Wang Ling, Wang Shengyao. A Novel Fruit Fly Optimization Algorithm for the Semiconductor Final Testing Scheduling Problem[J]. Knowledge-based Systems,2013,57(2):95-103.

[12] 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.

[13] Wang Lin, Shi Yuanlong, Liu Shan. An Improved Fruit Fly Optimization Algorithm and Its Application to Joint Replenishment Problems[J]. Expert Systems with Applications,2015,42(9):4310-4323.

[14] Marko M, Najdan V, Milica P, et al. Chaotic Fruit Fly Optimization Algorithm[J]. Knowledge-based Systems, 2015,89(C):446-458.

[15] Yuan Xiaofang, Dai Xiangshan, Zhao Jingyi, et al. On a Novel Multi-swarm Fruit Fly Optimization Algorithm and Its Application[J]. Applied Mathematics and Computation,2014,233(3):260-271.

编辑 刘 冰