基于维度分区的果蝇优化新算法

王友卫[1] 凤丽洲[2] 朱建明[1]
(中央财经大学信息学院 北京 100081)[1] (吉林大学计算机科学与技术学院 长春 130012)[2]

摘 要 为提高果蝇算法的收敛稳定性,提出了一种基于维度分区的果蝇优化新算法。将果蝇种群均分为两组:跟随果蝇和搜索果蝇。跟随果蝇在全局最优果蝇附近实现精细化局部搜索,而搜索果蝇则将位置向量的每个维度搜索范围划分为若干个区间,通过比较各个区间的最优位置来更新果蝇位置。为加快算法收敛速度,若某搜索果蝇在连续若干次迭代过程中均表现最差,则在当前最优果蝇位置附近产生该果蝇的新位置。针对8种典型函数的仿真实验表明:与传统算法相比,所提算法所需参数较少,收敛稳定性高,并且在收敛精度及收敛速度等方面具有明显优势。

关键词 果蝇算法,收敛稳定性,维度分区,全局最优果蝇,收敛精度
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.12.048

Novel Fruit Fly Optimization Algorithm Based on Dimension Partition

WANG You-wei[1] FENG Li-zhou[2] ZHU Jian-ming[1]
(School of Information, Central University of Finance and Economics, Beijing 100081, China)[1]
(College of Computer Science and Technology, Jilin University, Changchun 130012, China)[2]

Abstract In order to improve the convergence stability of fruit fly algorithm, a novel dimension partition based fruit fly optimization algorithm was proposed. The fruit fly population is divided into two groups: the following fruit flies and the searching fruit flies. A following fruit fly realizes the accurate local searching near the global best fruit fly, and a searching fruit fly divides each searching dimension of the position vector into several partitions and updates its position by comparing the performances of all partitions. In order to improve the convergence speed, if a searching fruit fly performs worst during several iterations, its new position will be generated near the global best fruit fly. The experimental results of 8 typical functions show that the proposed method needs fewer parameters, and has obvious advantages on convergence stability, convergence accuracy and convergence speed when comparing with traditional methods.

Keywords Fruit fly algorithm, Convergence stability, Dimension partition, Global best fruit fly, Convergence accuracy

1 引言

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]。与传统参数优化算法如粒子群优化(Particle Swarm Optimization, PSO)[3]、调和搜索(Harmony Search)[4]、遗传算法(Genetic Algorithm, GA)[5]等算法相比,FOA算法不仅实现简单、耗时小,而且所依赖的参数数目较少,能有效避免传统参数寻优算法中参数选取困难及参数对寻优结果影响较大的问题[6]。

目前,已涌现出许多基于FOA的改进算法。文献[7]提出了一种具有不同飞行半径的分群搜索策略,使得在搜索区间内果蝇的种群多样性大大增加。该算法针对不同的果蝇子群引入了不同的周期震荡函数,有效地提高了搜索精度、收敛速度和稳定性。但是,此算法中震荡半径的范围难以确定,且在震荡半径范围内生成的新位置随意性较大。文献[8]提出了一种动态双子群协同进化果蝇优化算法(DDSCFOA)。该算法根据群体的进化水平,动态地将整个种群划分为先进子群和后进子群。先进子群采用混沌算法在局部最优解邻域内进行精细的局部搜索,后进子群采用基本FOA算法进行全局搜索。该算法能较好地平衡局部搜索能力和全局搜索能力,但是后进子群寻优过程所使用的Logistic混沌映射难以获得均匀的随机分布效果;文献[9]提出利用修正型果蝇算法来优化广义回归神经网络(Generalized Regression Neural Network, GRNN),通过对原有果蝇算法添加逃脱参数以扩大搜索空间,但该算法并未对逃脱参数如何取值进行详细描述;为了避免传统FOA仅能对最优值为0的寻优问题进行求解,文献[6]对气味浓度参数进行归一化并且使用位置向量差代替原来的混沌搜索,但仍无法实现对负值参数的寻优;文献[10]使用果蝇位置向量直接计算适应度函数,去掉了气味浓度判定值及距离两个参数,该算法通过动态调整搜索半径提高算法的全局寻优能力,并且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。

2 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

到稿日期:2015-12-23 返修日期:2016-04-27 本文受国家自然科学基金项目(61272398)资助。
王友卫(1987-),男,博士,讲师,主要研究方向为机器学习、数据挖掘,E-mail:ywwang15@126.com;凤丽洲(1986-),女,博士生,主要研究方向为数据挖掘、社会化搜索;朱建明(1965-),男,博士,教授,博士生导师,主要研究方向为博弈论、信息安全。

位置，PAN 据此提出了果蝇优化算法[1]。具体步骤如下。

1. 初始化参数。设置种群中果蝇数量 N、最大迭代次数 T、搜索范围 $[x_m, x_M]$、果蝇的初始位置（$x_{\_axis}$，$y_{\_axis}$）、果蝇最优气味浓度 $Smell_{gb}$（$Smell_{gb}=-\infty$）等。

2. 按照式(1)随机产生每只果蝇的新位置（x_i，y_i）：

$$x_i = x_{\_axis} + RandomValue$$
$$y_i = y_{\_axis} + RandomValue \quad (1)$$

其中，RandomValue 为随机产生的搜索距离。

3. 计算果蝇对应的气味浓度判定值 s_i，如式(2)所示：

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

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

$$Smell_i = Fit(s_i) \quad (3)$$

5. 找出群体中对应的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），$Smell_b$。

6. 若 $Smell_b > Smell_{gb}$，则将位置（x_b，y_b）作为下一次产生新果蝇的初始位置。

7. 循环执行步骤 2—步骤 6，直到达到最大迭代次数，算法结束。

3 本文算法

IFFO 在搜索半径限定的范围内产生新的果蝇位置，但存在两个问题：1）寻优结果依赖搜索半径的大小；2）果蝇新位置的产生过程有很大的随机性，易获得不稳定的寻优结果。DDSCFOA 结合 Logistics 混沌映射从一定程度上避免了对搜索半径的依赖性，但仍面临以下问题：1）需要在迭代一定次数之后才会出现雪崩效应，而在此前所得的结果往往不是随机的；2）迭代序列的分布并不是完全均匀的，易出现“中间少两头多”的情况。

为了避免以上问题，本文先将果蝇均分成两个种群：搜索果蝇和跟随果蝇。其中，搜索果蝇通过控制果蝇在某个维度搜索范围内的遍历过程实现果蝇全局寻优搜索；跟随果蝇则在全局最优果蝇位置附近进行位置更新以实现精细化搜索。为了提高算法收敛的稳定性，提出了一种基于维度分区的搜索果蝇位置更新算法。

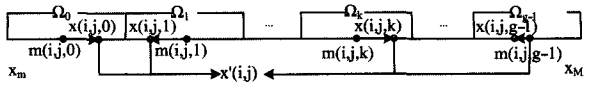

图 1 本文算法的维度分区过程示意

如图 1 所示，该算法将搜索果蝇 x_i 针对某一维度 $x(i,j)$ 的搜索范围均分成 g 个区域，分别记为 Ω_0，Ω_1，…，Ω_i，…，Ω_{g+1}。于是，每个区域的宽度 l 为：

$$l = \frac{x_M - x_m}{g} \quad (4)$$

其中，x_M，x_m 分别表示搜索范围的最大值、最小值。进一步，将每个区域的中间位置分别标记为 $m(i,j,k)$（$0 \leqslant k < g$）。当搜索果蝇 x_i 在第 j 维上搜索时，算法将在该维度每个区域的中心值 $m(i,j,k)$ 附近产生一个新位置 $x''(i,j,k)$，接着通过评估不同分区的 $x''(i,j,k)$ 值的表现确定果蝇 x_i 在第 j 维上的新位置 $x'(i,j)$。搜索果蝇位置更新过程如下。

输入：搜索果蝇 x_i 在第 j 维上的分量 x(i,j)，适应值函数 Fit

输出：位置更新后的搜索果蝇 x_i 在第 j 维上的分量 x'(i,j)

1. x'(i,j)= x(i,j)
2. for k:0:1:g−1
3. 按式(5)产生果蝇 x_i 在第 j 维的新位置 x''(i,j,k)：

$$x''(i,j,k) = x_m + \frac{1}{2} + (k-1) \times l + (-1)^{10000 \times Rand()} \times \frac{1}{2} \times Rand() \quad (5)$$

4. 设置 x_i 第 j 维分量值：x(i,j)= x''(i,j,k)；
5. 计算果蝇 x_i 对应的适应值 Fit(x_i)；
6. end for
7. 记录{Fit(x_i)}中最大值 Fit_b 及对应的 x''(i,j,q)(0≤q<g)；
8. if Fit_b>Fit(x_i)
9. x'(i,j)= x''(i,j,q)
10. end if

图 2 给出了本文算法的流程图。

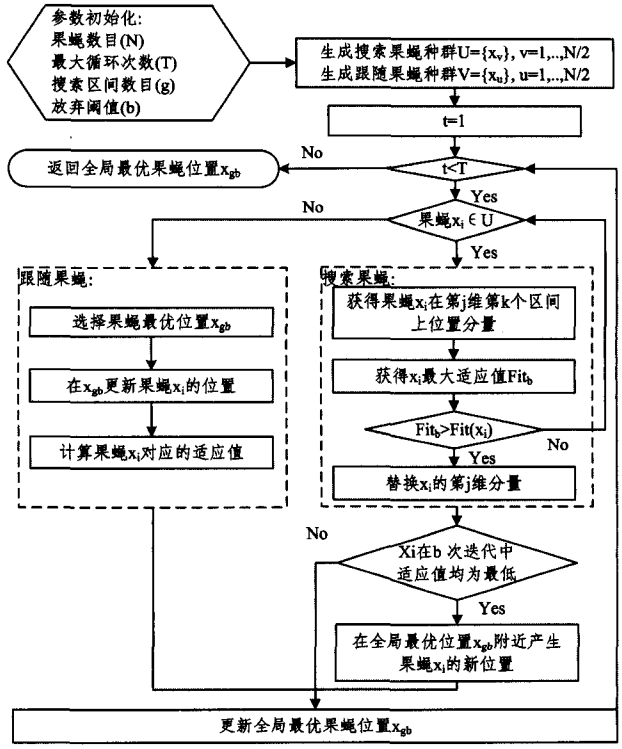

图 2 本文算法流程图

在此基础上给出算法的具体执行过程，如下所示。

1. 初始化果蝇种群 X(数量为 N)、搜索果蝇种群 U(U∈X，数量为 N/2)、跟随果蝇种群 V(V∈X，数量为 N/2)、最大迭代次数 T、向量维度 d、搜索范围$[x_{min}, x_{max}]$；
2. for $x_i \in X$，按照式(6)随机生成 x_i 的位置：

$$x(i,j) = x_{min} + (x_{max} - x_{min}) \times Rand(),\ 0 \leqslant j < d \quad (6)$$

其中，x(i,j)为果蝇 x_i 的第 j 维数值；

3. end for
4. for $x_i \in X$
5. 计算所有果蝇的适应值 Fit(x_i)；
6. end for
7. 获得全局最优果蝇并将其记为 x_{gb}；
8. 设置迭代次数 t=0；
9. while(未达到收敛状态)
10. for each $x_i \in X$
11. 按式(7)随机产生要更新的果蝇向量维度 j：

$$j = 10000 \times Rand()\ \%\ d \quad (7)$$

12. if $x_i \in U$
13. 按照搜索果蝇位置更新过程更新 x(i,j)值；
14. if x_i 连续 b 次对应的适应值 Fit(x_i)为全局最小值

15. 按照式(8)更新维度分量 x(i,j)值：

$$x(i,j)=x(gb,j)+\text{Rand}()\times(-1)^{\text{Rand}()\times 10000} \quad (8)$$

其中，x(gb,j)为 x_{gb}第 j 维分量；

16. end if

17. else

18. 按照下式计算跟随果蝇维度分量 x(i,j)值：

$$x(i,j)=x(gb,j)+\text{Rand}()\times(-1)^{\text{Rand}()\times 10000}$$

19. 计算 x_i 适应值 $\text{Fit}(x_i)$；

20. end if

21. end for

22. 更新全局最优果蝇 x_{gb}位置。

23. end while

上述过程中，步骤 12－步骤 16 给出了搜索果蝇的位置更新及果蝇淘汰过程。参考 IFFO 算法，本文在每次迭代中随机选择果蝇位置向量的某个维度进行更新。若某搜索果蝇连续多次对应的适应值较差，则认为该果蝇对于最终寻优结果是没有贡献的，并且此类果蝇的存在将影响算法的收敛速度。本文在步骤 14－步骤 16 中设置果蝇放弃阈值 $b(b=10)$，当某搜索果蝇 x_i 连续 b 次对应的适应值均为全局最小时，将在全局最优果蝇位置 x_{gb}附近产生该搜索果蝇的新位置。

4 实验结果与分析

4.1 实验设置

为测试不同算法性能，分别从全局搜索性能、时间复杂度、收敛精度及稳定性、参数依赖性等 4 个方面将本文与 DDSCFOA[8]，IFFO[10]，GOPSO[11]及 IHS[12]等优化算法进行比较。分别选取 3 个单峰基准函数(F_1-F_3)、5 个多峰基准函数(F_4-F_8)进行寻优测试[10]。不同函数的公式表示、变量取值范围、目标值及目标值对应的位置向量如表 1 所列。为公平起见，设置不同算法下的果蝇数量 $N=50$。为保证算法收敛，由本文执行过程知，最大迭代次数 T 应为一个足够大的值，故设置 $T=2000$。

表 1 不同函数公式表示、变量取值范围、目标值及目标值对应位置向量

函数	函数名	公式表示	变量取值范围	目标值	目标值对应的位置向量
F_1	step function	$f(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$[-100,100]$	0	$(0,0,\cdots,0)$
F_2	exponential problem function	$f(x)=-\exp(-0.5\sum_{i=1}^{n}x_i^2)+1$	$[-1,1]$	0	$(0,0,\cdots,0)$
F_3	dixon-price function	$f(x)=\sum_{i=n}^{n}i(2x_i^2-x_{i-1})^2+(x_1-1)^2$	$[-10,10]$	0	$(0,0,\cdots,0)$
F_4	alpine function	$f(x)=\sum_{i=1}^{n}\lvert x_i\sin(x_i)+0.1x_i\rvert$	$[-10,10]$	0	$(0,0,\cdots,0)$
F_5	schaffer function	$f(x)=\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{(1+0.001(x_1^2+x_2^2))^2}+0.5$	$[-100,100]$	0	$(0,0,\cdots,0)$
F_6	rastrigin function	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i))$	$[-5.2,5.2]$	0	$(0,0,\cdots,0)$
F_7	griewank function	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]$	0	$(0,0,\cdots,0)$
		$f(x)=\frac{\pi}{d}\{10\sin^2(\pi y_1)+\sum_{i=1}^{d-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_d-1)^2\}+\sum_{i=1}^{d}\mu(x_i,10,100,4)$			
F_8	generalized function	$y_i=1+\frac{1}{4}(x_i+1)$	$[-50,50]$	0	$(-1,-1,\cdots,-1)$
		$\mu(x_i,a,k,m)=\begin{cases}k(x_i-a)m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)m, & x_i<-a\end{cases}$			

4.2 针对维度分区数目 g 的选择

维度分区数目 g 取值的大小将直接影响本文算法的收敛时间及收敛精度。若 g 的取值过大，算法收敛精度较高，但是寻优速度较慢；若 g 的取值较小，算法收敛速度较快，但是收敛精度较低。为了测试 g 的取值对算法效果的影响，针对特定 g 值，定义算法针对函数 F_i 的收敛精度评价函数 $f(i,g)$如下：

$$f(i,g)=e^{-(\frac{1}{n}\sum_{j=1}^{n}|f_{i,j,g}-f_i|)} \quad (9)$$

其中，n 为针对 F_i 进行的实验次数，$f_{i,j,g}$为特定 g 值下函数 F_i 的第 j 次实验所得最优值，f_i 为函数 F_i 对应的目标值。进一步，统计当 g 在区间[2,20]取值时本文算法针对 8 个函数达到的收敛时间及收敛精度均值(分别记为 T_a，F_a)如图 3 所示。可见，当 g 逐渐增加时，T_a，F_a 均呈现出递增的趋势；而当 $g=6$ 时，F_a 值收敛于 1，说明此时算法已收敛至全局最优。因此，为了在保证算法收敛精度的同时尽可能地提高算法的执行效率，本文取 $g=6$ 并将其应用于后续实验中。

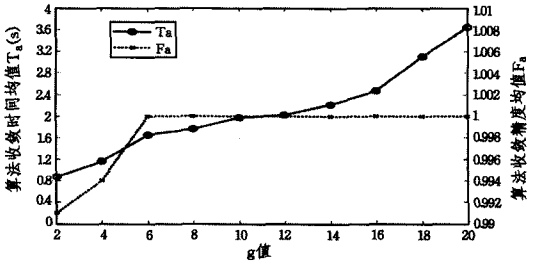

图 3 参数 g 的选择

4.3 全局搜索性能比较

一般而言，算法针对某个维度遍历得越完整，其越容易在该维度上找到最优值。为了避免算法陷入局部最优，IFFO(IHS)算法、DDSCFOA 算法分别使用 Rand 函数与 Logistic 混沌映射在果蝇向量某一维度空间中产生特定搜索范围内的随机值。由于该随机值具有较大的不确定性，因此极易导致

搜索“盲区”。为了比较不同算法的全局搜索性能，测试了在区间[0,1]中分别使用IFFO(IHS)算法、DDSCFOA算法及本文算法在果蝇位置向量某维度空间上进行10次搜索的结果。可见，相对于前两个算法，本文在10次搜索过程中进行了更多次的搜索尝试，并且搜索位置均匀分布在整个区间，说明本文基于分区的维度搜索策略能更好地保证算法的全局寻优效果，避免产生搜索“盲区”。

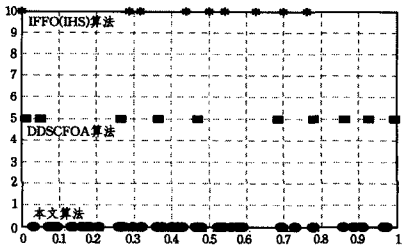

图4 全局搜索性能比较示意

4.4 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。本文算法的主要耗时过程包括：生成果蝇初始位置、更新果蝇位置、计算果蝇适应值3个阶段。给定迭代次数 T、果蝇数目 N、果蝇位置向量维数 d、分区数目 g，则本文生成果蝇初始位置过程的时间复杂度 T_1 表示如下：

$$T_1=O(d\times N) \tag{10}$$

更新果蝇位置及计算果蝇适应值过程的时间复杂度 T_2 表示为：

$$\begin{aligned}T_2&=O\left(T\times\left[\frac{N}{2}+\frac{N}{2}\times\Delta+\frac{N}{2}\times g\times\Delta\right]\right)\\&=O\left(T\times\left[\frac{N}{2}\times(1+(g+1)\times\Delta)\right]\right)\end{aligned} \tag{11}$$

其中，Δ 为计算适应值耗时。由于针对特定问题而言，g 和 Δ 均为固定值，因此本文算法的时间复杂度 T 可表示为：

$$\begin{aligned}T&=T_1+T_2\\&=O(d\times N)+O(T\times N)\\&=O((d+T)\times N)\end{aligned} \tag{12}$$

进一步地，表2列出了几种典型寻优算法对应的时间复杂度。由表2知，本文算法的计算复杂度与IFFO算法一致；由于DDSCFOA算法在每次迭代过程中都使用混沌映射产生新果蝇位置，因此时间复杂度最高。一般而言，由于存在 $d+T\ll d\times T$，因此本文及IFFO的时间复杂度较其他算法明显偏小。可见，本文随机选择一个维度实施果蝇位置更新对于降低算法的时间复杂度是十分有利的，并且本文直接通过搜索果蝇实施全局搜索，在扩大搜索范围的同时有效避免了混沌映射带来的迭代耗时。

表2 不同优化算法的时间复杂度对比

阶段时间复杂度	FOA算法	DDSCFOA算法	IFFO算法	GOPSO算法	IHS算法	本文算法
T_1	O(d×N)	O(d×N)	O(d×N)	O(d×N)	O(d×N)	O(d×N)
T_2	O(d×N×T)	O(d×N×T×t)	O(N×T)	O(d×N×T)	O(d×N×T)	O(N×T)
T	O(d×N×T)	O(d×N×T×t)	O((d+T)×N)	O(d×N×T)	O(d×N×T)	O((d+T)×N)

4.5 收敛精度及稳定性比较

为了测试不同算法的收敛精度及稳定性，针对每种优化算法，令维度 d 分别取200,400,600,800,1000，并分别针对每个 d 值进行100次实验，最后按照式(13)计算100次实验所得函数的最优值均值 E 和平均方差 D：

$$\begin{aligned}E&=\frac{1}{500}\sum_{i,d}f_{i,d}\\D&=\frac{1}{500}\sum_{i,d}|f_{i,d}-E|\end{aligned} \tag{13}$$

其中，i 表示针对每个 d 值进行的实验次数，$i\in[1,100]$，$f_{i,d}$ 表示当维度为 d 时第 i 次实验所得函数的最优值。表3中符号±前、后的数值分别为按照式(13)计算出的每种函数对应的 E、D 值。由表3知，DDSCFOA算法由于直接使用味道浓度作为选择全局最优果蝇的判定标准，因此无法实现针对负值参数的寻优，导致算法收敛精度普遍偏低；IFFO与GOPSO表现相近，但鉴于这两种算法的寻优结果分别受到搜索半径及种群分布的影响，因此在处理多极值函数方面皆稍逊于IHS与本文算法；本文所得 E 值与IHS算法近似，但在处理多极值寻优问题时所得 D 值相对较小，说明本文算法能保证算法的全局搜索精度，同时可有效提高算法的稳定性。

表3 不同算法的收敛精度及稳定性比较

函数	DDSCFOA算法	IFFO算法	GOPSO算法	IHS算法	本文算法
F_1	2.35E−5± 3.26E−5	2.32E−6± 5.87E−7	3.32E−5± 5.24E−8	6.74E−6± 5.17E−6	6.23E−6± 7.33E−6
F_2	5.56E−4± 8.11E−4	2.57E−7± 1.22E−6	8.16E−7± 7.61E−6	5.35E−7± 2.52E−5	6.74E−7± 8.58E−5
F_3	6.23E−7± 7.93E−7	6.36E−7± 3.45E−7	5.32E−6± 4.68E−6	8.63E−7± 4.07E−6	7.33E−6±2.10E−7
F_4	2.96E−5± 3.55E−5	1.79E−5± 4.38E−4	1.71E−5± 5.22E−7	7.53E−6±2.73E−6	4.11E−6± 5.92E−7
F_5	4.82E−4±2.72E−9	5.55E−6± 8.32E−6	8.23E−6± 6.76E−5	8.22E−7± 5.03E−5	5.31E−7± 7.25E−8
F_6	2.46E−5±1.43E−7	7.02E−6± 1.62E−6	4.77E−5± 4.87E−5	4.69E−7± 4.09E−6	4.62E−7± 2.14E−7
F_7	5.34E−4±5.87E−8	6.51E−6± 7.35E−6	5.82E−7± 4.33E−6	7.95E−7± 7.63E−5	5.82E−7± 7.37E−7
F_8	1.89E−3± 9.59E−4	8.25E−5± 3.77E−6	9.34E−5± 3.64E−6	1.24E−6± 8.65E−4	2.18E−7± 5.76E−7

4.6 参数依赖性比较

性能良好的优化算法应能有效避免初始参数的影响。为了比较不同算法对于初始参数的依赖性，给出了几种典型优化算法所需的参数，如表4所列。可见，本文与FOA算法对应参数数目最少；DDSCFOA算法与IFFO算法所需参数均为5，它们分别在FOA算法的基础上增加了对混沌映射迭代次数、搜索半径的依赖性；与其他算法相比，GOPSO算法与IHS所需参数最多，分别为6和9，说明这两种算法对参数的依赖性较强，算法易获得不稳定的寻优结果。

表 4　算法参数依赖性比较

寻优算法所需参数	FOA 算法	DDSCFOA 算法	IFFO 算法	GOPSO 算法	IHS 算法	本文算法
群体规模 N	√	√	√	√	√	√
最大迭代次数 T	√	√	√	√	√	√
特征维数 D	√	√	√	√	√	√
搜索范围最大值 x_M 及最小值 x_m	√	√	√	√	√	√
混沌映射迭代次数 t_l		√				
搜索半径最大值 r_M 及最小值 r_m			√			
粒子最大速度 v_M 及最小速度 v_m				√		
认知学习因子 c_1 及社会化搜索因子 c_2				√		
调和内存大小 H_s					√	
调和内存考虑率 H_r					√	
调节率 P					√	
调整率的最大值 P_M 及最小值 P_m					√	
调整宽度的最小值 bw_{min} 和最大值 bw_{max}					√	

结束语　针对传统果蝇优化算法稳定性不高且易陷入局部最优的问题，提出了一种基于维度分区的果蝇寻优新算法，其主要特点如下：1)为提高算法的全局搜索能力，将果蝇种群均分为两类并将它们分别称为跟随果蝇和搜索果蝇；2)为提高寻优结果的稳定性，在每次迭代过程中，将搜索果蝇在每个维度上的搜索范围均分为若干个区间，通过评估各个区间的表现产生果蝇的新位置；3)为加快算法的收敛速度，搜索果蝇将舍弃那些对应适应值最小的位置，并转而飞向当前全局最优位置。针对 8 种不同类型函数的寻优结果表明，本算法在收敛精度、稳定性及参数依赖性方面较传统优化算法具有明显的优势。

参 考 文 献

[1] Pan W C. Using fruit fly optimization algorithm optimized general regression neural network to construct the operating performance of enterprises model [J]. Journal of Taiyuan University of Technology (Social Science Edition), 2011, 29(4): 1-5(in Chinese)
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: 69-74

[3] Brahim, Chellali B. Power quality enhancement using shunt active power filter based on particle swarm optimization [J]. Journal of Applied Sciences, 2011, 11(22): 3725-3731

[4] Geem Z W, Kim J H, Loganathan G V. A new heuristic optimization algorithm: harmony search [J]. Simulation, 2001, 76 (2): 60-68

[5] Cha S H, Tappert C C. A genetic algorithm for constructing compact binary decision trees [J]. Journal of Pattern Recognition Research, 2009, 4(1): 1-13

[6] Niu J, Zhong W, Liang Y, 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

[7] Zhu Z T, Guo X, Li W. New fruit fly optimization algorithm research [J]. Computer Engineering and Applications, 2015 (in Chinese)
朱志同, 郭星, 李炜. 新型果蝇优化算法的研究[J]. 计算机工程与应用, 2015

[8] Han J Y, Liu C Z, Wang L G. Dynamic double groups cooperate fruit fly optimization algorithm [J]. Pattern Recognition and Artifical Intelligence, 2013, 26(11): 1057-1067(in Chinese)
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] Wang Y B, Nie N N, Wang M Z, et al. Mine tailing facilities safety evaluation of GRNN optimized by modifies fruit fly algorithm [J]. Computer Engineering, 2015, 41(4): 267-272(in Chinese)
王英博, 聂娜娜, 王铭泽, 等. 修正型果蝇算法优化 GRNN 网络的尾矿库安全预测[J]. 计算机工程, 2015, 41(4): 267-272

[10] Pan Q K, Sang H Y, Duan J H, et al. An improved fruit fly optimization algorithm for continuous function optimization problems [J]. Knowledge-Based Systems, 2014, 62(5): 69-83

[11] Wang Y W, Liu Y N, Zhu X D. Two-step based hybrid feature selection method for spam filtering [J]. Journal of Intelligent & Fuzzy Systems, 2014, 27(6): 2785-2796

[12] Mahdavia M, Fesangharyb M, Damangir E. An improved harmony search algorithm for solving optimization problems [J]. Applied Mathematics and Computation, 2007, 188(2): 1567-1579

(上接第 263 页)

李进, 张江华. 基于碳排放与速度优化的带时间窗车辆路径问题[J]. 系统工程理论与实践, 2014, 34(12): 3063-3072

[6] Dong Wen-yong, Zhang Deng-yi, Zhong Wei-cheng. The Simulation Optimization Algorithm Based on the Ito Process[M]// Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques, 2007: 115-124

[7] Dong Wen-yong, Lei Ming, Yu Rui-guo. A New Evolutionary Algorithms for Global Numerical Optimization Based on Ito Process[M] // Computational Intelligence and Intelligent Systems. Springer Berlin Heidelberg, 2010: 57-67

[8] Dong Wen-yong, Yu Rui-guo, Lei Ming. Merging the Ranking and Selection into ITO Algorithm for Simulation Optimization [M] // Computeational Intelligence and Intelligent Systems. Springer Berlin Heidelberg, 2010: 87-96

[9] Dong Wen-yong, Zhang Deng-yi, Li Yuan-xiang. The multi-objective ITO algorithms[C]// Proceedings of the 2nd International Conference on Advances in Computation and Intelligence. Springer-Verlag, 2007: 53-61

[10] Dong Wen-yong, Zhang Wen-sheng, Yu Rui-guo. Convergence and Runtime Analysis of ITO Algorithm for One Class of Combinatorial Optimization[J]. Chinese Journal of Computers, 2011, 34(4): 636-646(in Chinese)
董文永, 张文生, 于瑞国. 求解组合优化问题伊藤算法的收敛性和期望收敛速度分析[J]. 计算机学报, 2011, 34(4): 636-646

[11] Yi Yun-fei, Cai Yong-le, Dong Wen-yong, et al. Improved ITO Algorithm for Solving the CVRP[J]. Computer Science, 2013, 40(5): 213-216(in Chinese)
易云飞, 蔡永乐, 董文永, 等. 求解带容量约束的车辆路径问题的改进伊藤算法[J]. 计算机科学, 2013, 40(5): 213-216

[12] Ubeda S, Arcelus F J, Faulin J. Green logistics at Eroski: A case study[J]. International Journal of Production Economics, 2011, 131(1): 44-51