•信息工程•

DOI:10.15961/j.jsuese.201600671

基于群密度的改进果蝇优化算法及在异常检测中的应用

王友卫[1],朱建明[1],凤丽洲[2]

(1.中央财经大学 信息学院,北京 100081;2.天津财经大学 理工学院,天津 300222)

摘　要:针对传统果蝇算法面临的收敛稳定性差、难以协调全局搜索及局部搜索能力等缺点,提出一种基于群密度的改进果蝇优化算法。首先,借鉴现有算法的优势,将果蝇种群分为搜索果蝇和跟随果蝇,并分别使用两类果蝇进行全局化搜索与局部精细化搜索。然后,为提高算法全局搜索的稳定性,在每次迭代过程中使用基于最优区间回避的分区采样策略更新搜索果蝇的位置;该策略在每次迭代过程中获得表现最优的若干只果蝇以构造最优果蝇组,根据最优果蝇组中果蝇个体在每个维度上的取值范围确定最优区间,并通过对最优区间外的其他区间分区采样以确定搜索果蝇的新位置。最后,为协调算法的全局搜索能力与局部搜索能力,引入群密度的概念,通过计算果蝇群密度并结合相关阈值实现不同种群规模的动态调整。针对典型测试函数的实验结果表明,基于最优区间回避的分区采样策略相对于传统随机函数具有更强的全局优化性能。与传统优化算法相比,本文算法在保证收敛速度的同时获得了较高的寻优精度及稳定性,在综合性能上得到明显提升。在KDDcup99数据集上的异常检测仿真实验结果表明,本文基于分区采样及群密度的果蝇优化算法能有效避免局部最优,在获取异常检测分类器的重要参数最佳取值方面起到一定作用。

关键词:果蝇算法;收敛稳定性;全局搜索;局部搜索;异常检测

中图分类号:TP301.6　　**文献标志码**:A　　**文章编号**:2096-3246(2017)05-0127-08

Improved Fruit Fly Optimization Algorithm Based on Population Density and Its Application in Anomaly Detection

WANG Youwei[1], *ZHU Jianming*[1], *FENG Lizhou*[2]

(1.School of Info., Central Univ. of Finance and Economics, Beijing 100081, China;
2.School of Sci. and Eng., Tianjin Univ. of Finance and Economics, Tianjin 300222, China)

Abstract: To address the problems that the traditional fruit fly algorithms cannot steadily bad converge,and effectively balance the global and local searching ability,a novel fruit fly optimization algorithm based on population density was proposed.Firstly,by utilizing the advantages of existing methods,the fruit flies were divided into the searching fruit flies and the following fruit flies,which were then used for global searching and local searching,respectively.Secondly,in order to improve the stability of global searching process,the partition sampling strategy based on optimal interval avoidance was used to update the positions of searching fruit flies in each iteration process.The strategy selected the fruit flies with the best performances in each iteration to construct the optimal fruit fly group,and determined the optimal interval according to the ranges of the fruit flies in each dimension.Then,the new positions of the searching fruit flies were determined by sampling the interval except for the optimal interval.Finally,in order to balance the global and local searching ability,the conception of population density was introduced,and the dynamic population size adjustment of different types of fruit flies was achieved by thresholding the population density.The experimental results of typical test functions showed that the partition sampling strategy based on optimal interval avoidance achieved higher global optimization ability compared to traditional rand functions.Compared to traditional optimization algorithms,the proposed algorithm obtained high optimization accuracy and stability while guaranteeing the convergence speed,achieving obvious improvements on comprehensive performances.The simulation results of the anomaly detection showed that,the fruit fly algorithm based on the partition sampling and population density can avoid local optimum

收稿日期:2016 – 07 – 03

基金项目:北京市自然科学基金资助项目(4174105);中央财经大学学科建设基金资助项目(2016XX02);国家自然科学基金重点支持项目–NSFC-浙江两化融合联合基金项目资助(U1509214)

作者简介:王友卫(1987—),男,讲师,博士. 研究方向:信息安全;机器学习;群体智能. E-mail:wyw4966198@126.com

effectively,and is effective in obtaining the optimal values of important parameters of the anomaly detection classifier.

Key words: fruit fly algorithm;convergence stability;global searching;local searching;anomaly detection

启发式优化算法具有智能性、并行性、渐进性、随机性等基本特点，近年来引起了众多学者的广泛研究，目前已成功应用于科学、工程和金融等领域。然而，传统启发式优化算法（如粒子群算法[1]、遗传算法[2]、调和搜索算法[3]等）在参数依赖性、计算复杂性、收敛速度、优化精度等方面面临许多问题。2011年，潘文超通过模拟果蝇觅食行为提出果蝇优化算法（fruit fly optimization algorithm, FOA）[4–5]。与其他算法相比，FOA具有参数少、计算简单、容易理解等优点，因此已广泛应用于参数优化[6]、PID控制器设计[7]、电力负荷预测[8]等诸多领域。

FOA算法在每次迭代过程中单纯以全局最优点为目标导向，极易陷入局部极值，导致收敛精度较低。文献[9]对FOA算法进行改进，通过动态调整搜索半径提高算法的全局寻优能力，但不足之处在于寻优结果对于搜索半径的依赖性较强，因此算法全局搜索能力较差。针对传统算法极易陷入局部最优的问题，文献[10]提出一种基于近郊区和远郊区的果蝇优化算法，该算法将果蝇在每个维度上的搜索范围分为近郊区和远郊区，通过引入局部最优导向因子动态调整果蝇在不同区域的搜索强度；该算法避免了寻优结果对搜索半径的依赖，但全局搜索结果的稳定性较差。文献[11]通过计算果蝇的适应值大小决定是否进行全局搜索，算法全局优化效果较好，但不足之处是果蝇产生的新位置随机性大，难以有效协调算法的收敛速度及稳定性。文献[12]在果蝇位置更新时引入随机因子，指出使用混沌映射能有效改善算法的优化效果，但果蝇新位置的产生受当前位置影响，因此易导致不稳定的收敛结果。文献[13]引入双子群及维度分区思想，较好地平衡了算法的全局搜索能力和局部搜索能力，但不足之处在于不同类型果蝇种群的规模固定使得算法的灵活性较差。

综上，现有的针对FOA的改进算法普遍面临以下问题：1）在全局搜索时未考虑求解区间内特定区域的特殊性，导致果蝇新位置产生过程具有一定的盲目性，使得算法优化结果不稳定；2）未充分考虑实施全局搜索及局部搜索的临界状态，难以动态协调算法的局部搜索及全局搜索强度。为此，本文对传统算法做出改进，引入群密度的概念，实现了一种基于群密度的果蝇优化算法。具体而言，本文主要贡献如下：1）为提高算法稳定性，提出一种基于最优区间回避的分区采样策略；2）为协调算法的全局搜索能力和局部搜索能力，提出一种基于群密度的种群规模动态调整策略；3）通过典型测试函数及异常检测仿真实验验证了本文在收敛精度、收敛速度、稳定性等方面的有效性。

1 传统果蝇优化算法

与粒子群算法、遗传算法等类似，果蝇算法以随机值作为初始位置，在迭代寻优过程中所有个体都飞向到上一代最优个体，并通过在最优个体附近搜索寻找全局最优解。算法主要执行步骤如下：

输入：果蝇种群$X=\{\boldsymbol{x}_i\}$（$1<i\leq N$, N为种群中果蝇数量）、搜索范围$[x_{\min}, x_{\max}]$、最大迭代次数T。

输出：果蝇最优位置$(x_{\mathrm{b}1}, x_{\mathrm{b}2})$、果蝇最优气味浓度$Smell_{\mathrm{gb}}$。

步骤1：初始化参数。具体包括：果蝇数量N、最大迭代次数T、果蝇最优位置$(x_{\mathrm{b}1}, x_{\mathrm{b}2})$、果蝇最优气味浓度$Smell_{\mathrm{gb}}$（$Smell_{\mathrm{gb}}=-\infty$）等。

步骤2：更新每只果蝇$\boldsymbol{x}_i$的位置(x_{i1}, x_{i2})：

$$x_{i1}=x_{\mathrm{b}1}+\mathrm{Rand}(), x_{i2}=x_{\mathrm{b}2}+\mathrm{Rand}() \tag{1}$$

其中，Rand( )产生一个(0, 1)区间内的随机数。

步骤3：求得$\boldsymbol{x}_i$对应的气味浓度判定值s_i，如式（2）所示：

$$d_i=\sqrt{x_{i1}{}^2+x_{i2}{}^2}, s_i=\frac{1}{d_i} \tag{2}$$

步骤4：根据适应度函数Fit求得果蝇$\boldsymbol{x}_i$的气味浓度$Smell_i$：

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

步骤5：将气味浓度最大的果蝇对应的位置及气味浓度值分别记为$(x_{\mathrm{m}1}, x_{\mathrm{m}2})$、$Smell_{\mathrm{m}}$。

步骤6：若$Smell_{\mathrm{m}}>Smell_{\mathrm{gb}}$，则执行：

$$x_{\mathrm{b}1}=x_{\mathrm{m}1},\quad x_{\mathrm{b}2}=x_{\mathrm{m}2} \tag{4}$$

步骤7：循环执行步骤2～6直至达到最大迭代次数。

2 本文算法

2.1 算法描述

为提高全局搜索能力，文献[10]引入近郊区和远郊区的概念，避免了优化结果对搜索半径的依赖，但在远郊区产生的果蝇新位置仍有较大的随机性。文献[13]综合考虑传统果蝇算法及人工蜂群（artificial bee colony, ABC）算法的优点和不足，提出搜索果蝇和跟随果蝇的概念；基于此，该文提出一种基于维度分区的搜索果蝇位置更新算法。但该算法在分区过程中未考虑果蝇最优位置的特殊性，易导致局部最优位置被重复搜索，继而影响算法的收敛速度及稳

定性。为此, 本文提出一种基于最优区间回避采样的搜索果蝇位置更新策略。该策略首先在每次迭代过程中获得表现最优的前m只果蝇构造最优果蝇组, 然后根据最优果蝇组中果蝇在每个维度上的取值范围确定最优区间, 最后结合文献[13]对最优区间外的其他区间进行分区采样以确定搜索果蝇的新位置。

如图1所示, 给定最优果蝇组$GB=\{\boldsymbol{x}_i'\}$($1<i\le m$)、搜索区间$[x_{\min}, x_{\max}]$、步长l及维度j, 按照式(5)计算x_i'在第j维上的取值x_{ij}'的最小值w_j、最大值z_j分别为:

$$w_j=\min_{x_i'\in GB}(x_{ij}'),z_j=\max_{x_i'\in GB}(x_{ij}') \tag{5}$$

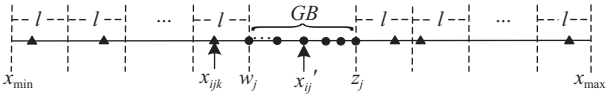

图 1　基于最优区间回避的分区采样过程示意

Fig.1　Process of optimal interval avoidance based partition sampling

为此, 将区间$[w_j, z_j]$定义为果蝇在第j维上的最优区间。由文献[13]可知, 跟随果蝇的精细化搜索过程已涵盖针对最优区间的搜索, 因此, 本文中搜索果蝇将回避针对区间$[w_j, z_j]$的搜索, 而只对区间$[x_{\min}, w_j)$、$(z_j, x_{\max}]$进行分区采样, 以提升算法的全局搜索稳定性。给定采样次数g(本文取g=10)、果蝇适应值计算函数Fit, 以最大化问题为例给出基于最优区间回避分区采样的搜索果蝇位置更新过程:

输入: 搜索果蝇$\boldsymbol{x}_i$在维度j上的分量x_{ij}、适应值函数Fit、采样次数g、最优区间$[w_j, z_j]$。

输出: 位置更新后的搜索果蝇$\boldsymbol{x}_i'$。

1. 初始化$\boldsymbol{x}_i$对应的最大适应值$F_{\rm b}$、$\boldsymbol{x}_i$在第j维上的最优取值$t_{\rm b}$及采样步长l:

$$\begin{cases} F_{\rm b}=F(\boldsymbol{x}_i),\ t_{\rm b}=x_{ij}, \\ l=(x_{\max}-x_{\min}-(z_j-w_j))/g \end{cases} \tag{6}$$

2. for k=1:1:g

3. $x_{ijk}=x_{\min}+(k-1)\times l+l\times{\rm Rand}()$ (7)

4. if $x_{ijk}>w_j$

5. $x_{ijk}=x_{ijk}+(z_j-w_j)$ (8)

6. end if

7. $x_{ij}=x_{ijk},\ F_{ijk}=Fit(x_i)$ (9)

8. if $F_{ijk}>F_{\rm b}$

9. $F_{\rm b}=F_{ijk}, t_{\rm b}=x_{ijk}$

10. end if

11. end for

12. $x_{ij}=t_{\rm b},\ \boldsymbol{x}_i'=\boldsymbol{x}_i$

文献[13]使用搜索果蝇和跟随果蝇协调种群的全局搜索能力和局部搜索能力, 但两类种群规模固定, 难以根据算法收敛情况实现种群规模的动态调整。为此, 定义群密度概念, 通过计算跟随果蝇种群的收敛情况动态调整两类种群的规模。给定搜索果蝇种群$U=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_P\}$($P$为搜索果蝇数目)、跟随果蝇种群$V=\{\boldsymbol{x}_{P+1}, \boldsymbol{x}_{P+2}, \cdots, \boldsymbol{x}_N\}$, 群密度$\sigma$可通过式(10)获得:

$$\sigma=\sqrt{\frac{1}{(x_{\max}-x_{\min})\times(N-P)\times D}\sum_{j=1}^{D}\sum_{i=P+1}^{N}\left|x_{ij}-\mu_j\right|^2} \tag{10}$$

式中, $\mu_j=\dfrac{1}{N-P}\sum_{i=P+1}^{N}x_{ij}$, j为随机选择的某一维度, D为果蝇位置向量维度。可见, 群密度表示跟随果蝇位置向量的标准差, 反映了跟随果蝇群体的聚集程度。进一步引入群密度阈值$th_{\rm u}$、$th_{\rm l}$($th_{\rm u}>th_{\rm l}$), 在每次迭代过程中, 将跟随果蝇的群密度值σ与$th_{\rm u}$、$th_{\rm l}$比较以实现不同类型种群规模的动态调整: 当$\sigma>th_{\rm u}$时, 跟随果蝇分布过于分散, 此时将部分搜索果蝇转化为跟随果蝇, 以提高算法的局部搜索能力; 当$\sigma<th_{\rm l}$时, 跟随果蝇分布过于集中, 此时将部分跟随果蝇转化为搜索果蝇, 以提高算法的全局搜索能力。基于此, 给出本文基于群密度改进果蝇优化算法的执行步骤如下:

输入: 果蝇种群$X=\{\boldsymbol{x}_i\}$($1<i\le N$, N为种群中果蝇数量); 搜索范围$[x_{\min}, x_{\max}]$; 最大迭代次数T; 群密度阈值$th_{\rm u}$、$th_{\rm l}$; 位置向量维度D; 搜索半径r最大值$r_{\max}$、最小值$r_{\min}$。

输出: 果蝇全局最优位置向量$\boldsymbol{x}_{\rm bp}$。

1. 初始化果蝇种群X、搜索果蝇种群$U=\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{P_m}\}$、跟随果蝇种群$V=\{\boldsymbol{x}_{P_m+1}, \boldsymbol{x}_{P_m+2}, \cdots, \boldsymbol{x}_m\}$、最优适应值$F_{\rm b}$=+∞。

2. for $\boldsymbol{x}_i\in X$, 按照式(11)随机生成$\boldsymbol{x}_i$的位置:

$$x_{ij}=x_{\min}+(x_{\max}-x_{\min})\times{\rm Rand}(),\ 0\le j<D \tag{11}$$

式中, x_{ij}为果蝇$\boldsymbol{x}_i$的第j维数值。

3. end for

4. 设置迭代次数t=0。

5. while($t<T$)

6. 获得最优果蝇组GB。

7. for each $\boldsymbol{x}_i\in U$

8. 按照式(12)产生搜索果蝇位置向量维度j:

$$j=\lfloor 10\,000\times{\rm Rand}()\rfloor \bmod D \tag{12}$$

9. 利用基于最优区间回避分区采样的搜索果蝇位置更新策略确定x_{ij}值。

10. end for

11. for each $\boldsymbol{x}_i\in V$

12. 按照式(12)产生跟随果蝇位置向量维度j。

13. 按照式(13)计算跟随果蝇维度分量x_{ij}值:

$$x_{ij}=x_{bj}+r\times(-1)^{\lfloor \mathrm{Rand}()\times 10\,000\rfloor}\times \mathrm{Rand}() \quad (13)$$

式中，x_{bj}为$\boldsymbol{x}_{bp}$的第j维分量。

14. end for

15. 更新全局最优果蝇$\boldsymbol{x}_{bp}$及搜索半径r。

16. 按照式(10)计算搜索果蝇群密度σ。

17. if $\sigma>th_u$,

18. $P=P_m$。

19. end if

20. if $\sigma<th_l$

21. $P=N-P_m$, $r=r_{max}$。

22. end if

23. t++。

24. end while

其中，P_m为搜索果蝇数目最小值，r为动态搜索半径，更新过程如式(14)所示[9]：

$$r=r_{max}\times\exp\left(\log\left(\frac{r_{min}}{r_{max}}\right)\times\frac{t}{t_{max}}\right) \quad (14)$$

式中，r_{min}=0.001，r_{max}=(x_{max}−x_{min})/2。

可见，与IFOA[11]、cFOA[12]、DPFOA[13]等算法相比，本文基于群密度改进果蝇优化算法具有两大特点：1)第9行中，通过实施基于最优区间回避分区采样的搜索果蝇位置更新策略，避免传统Rand函数、混沌映射函数等方法因随机性较大而带来的收敛结果不稳定的问题；2)第16～22行中，根据果蝇群密度大小实现不同类型果蝇种群规模的动态调整，在保证精细化搜索的同时提高算法全局搜索能力。

2.2 时间复杂度分析

本文基于群密度改进果蝇优化算法中，第1～4行为种群初始化过程，对应的时间复杂度为：

$$TC_1=\mathrm{O}(D\times N) \quad (15)$$

第6～24行为果蝇种群位置更新及群密度计算过程，对应时间复杂度为：

$$TC_2=\mathrm{O}((m\times N+P\times g+(N-P)+D\times N)\times T) \quad (16)$$

其中，P、N分别为搜索果蝇数目及果蝇总数目，m为最优果蝇组中果蝇数目，g为分区采样次数。在此基础上，得到本文算法时间复杂度为：

$$TC=TC_1+TC_2=\mathrm{O}(D\times N+(m\times N+P\times g+(N-P)+D\times N)\times T) \quad (17)$$

由于m、g均为常数，式(17)可化简为：

$$TC=\mathrm{O}(N\times D\times T) \quad (18)$$

用类似的方法分析得FOA、IFFO、IFOA、cFOA及DPFOA对应的时间复杂度分别为TC_{FOA}=O(D×N×T)、TC_{IFFO}=O((D+T)×N)、TC_{IFOA}= O(D×N×T)、TC_{cFOA}= O(D×N×T)、TC_{DPFOA}= O((D+T)×N)。可见，本文算法时间复杂度虽高于IFFO、DPFOA算法，但与FOA、IFOA、cFOA等算法相近，这说明本文算法并未因分区采样及群密度计算过程带来显著的计算开销。

3 实验结果与分析

3.1 实验环境

实验硬件环境为Intel core i7-6700 处理器、8.00 G内存，软件环境为Win7的64位操作系统、Matlab 2013。如表1所示，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)进行测试[9,13]。选用的对比算法包括FOA[4–5]、IFFO[9]、IFOA[11]、cFOA[12]及DPFOA[13]。设置不同算法对应的果蝇数量N=40、最大迭代次数T=5 000。

表 1 不同函数公式表示、目标值及目标位置向量

Tab.1 Formulas, target values and target position vectors of different functions

函数	函数名	公式表示	目标值	目标位置向量
F_1	Fabs	$f(\boldsymbol{x})=\sum_{i=1}^{D}\lvert x_i+1\rvert$	0	(−1, −1, ⋯, −1)
F_2	Rosenbrock	$f(\boldsymbol{x})=\sum_{i=1}^{D}\left(100\left(x_i-x_i^2\right)^2+\left(x_i-1\right)^2\right)$	0	(1, 1, ⋯, 1)
F_3	Quartic	$f(\boldsymbol{x})=\sum_{i=1}^{D}ix_i^4$	0	(0, 0, ⋯, 0)
F_4	Rastrigin	$f(\boldsymbol{x})=\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	0	(0, 0, ⋯, 0)
F_5	Salomon	$f(\boldsymbol{x})=1-\cos\left(2\pi\sqrt{\sum_{i=1}^{D}x_i^2}\right)+0.1\sqrt{\sum_{i=1}^{D}x_i^2}$	0	(0, 0, ⋯, 0)
F_6	Generalized Penalized	$f(\boldsymbol{x})=\frac{\pi}{d}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\cdot\left[1+10\sin^2(\pi y_{i+1})\right]+(y_D-1)\right\}+\sum_{i=1}^{D}\mu(x_i,10,100,4)$，$y_i=1+\frac{1}{4}(x_i+1)$，$\mu(x_i,a,k,m)=\begin{cases}k(x_i-a)m, x_i>a;\\0, -a\le x_i\le a;\\k(-x_i-a)m, x_i<-a\end{cases}$	0	(−1, −1, ⋯, −1)

3.2　群密度阈值th_u、th_l的选择

群密度阈值th_u、th_l的取值大小将影响算法的全局搜索能力和局部搜索能力：若当前密度大于th_u，应增加跟随果蝇的数量以提高算法局部搜索能力；若当前密度小于th_l，应增加搜索果蝇的数量以提高算法的全局搜索能力。本文以果蝇群体初始化密度均值为基础计算th_u，即

$$th_u=\frac{\theta}{t_e}\sum_{i=1}^{t_e}\sigma_i \tag{19}$$

式中：t_e为实验次数，取t_e=100；θ为群密度权重控制因子，取θ=0.9；σ_i为在第i次实验中果蝇群体初始化状态对应的群密度。为获得较为通用的th_l值，除3.1节所给出的6个测试函数外，结合文献[9]中的2个单峰函数（Sphere函数与Step函数）及2个多峰函数（Griewank函数与Whitley函数）确定最佳th_l。令维度D分别取2、4、6、8、10，当$th_l\in(0, th_u)$以特定步长（0.01）递增时统计不同函数对应的最佳适应值均值（F_a）及达到收敛状态对应的迭代次数均值（N_{ca}），结果如图2所示。

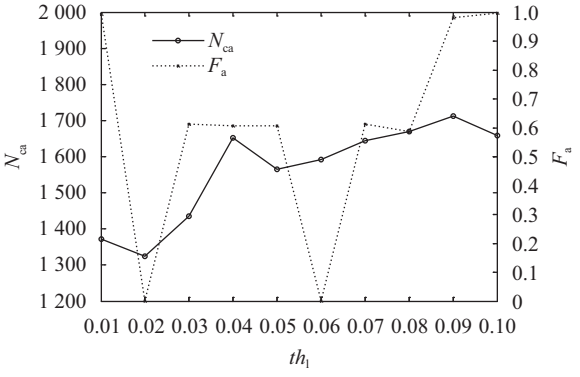

图 2　群密度阈值的选择

Fig.2　Selection of population density thresholds

由图2可知：当th_l逐渐增加时，N_{ca}值呈增大趋势，原因在于算法过度强调全局搜索性能而弱化了精细化搜索过程；进一步可知，F_a值随th_l增加呈现出一定的波动性，但当$th_l\in[0.02, 0.08]$时，F_a值相对较小，说明算法此时能较好地协调全局搜索及局部搜索强度。因此，为保证收敛精度的同时尽可能地减少算法达到收敛所需的迭代次数，本文取th_l=0.02并将其应用于后续实验中。

3.3　全局优化能力比较

Rand函数[11]、Logistic函数[11]与Circle函数[12]为现有的3种主要全局优化函数。为验证不同方法的全局优化能力，分别使用Rand函数、Logistic函数、Circle函数及本文基于最优区间回避采样的搜索果蝇位置更新策略对复杂的多极值函数F_4～F_6进行测试，计算100次全局搜索情况下搜索果蝇对应的适应值均值随迭代次数的变化情况，结果如图3所示。由图3可知，随着迭代次数的增加，基于最优区间回避采样的搜索果蝇位置更新策略对应结果下降明显，且在50次迭代后获得适应值均值极小值（0.40）；其他方法对应的极小值均大于2，且达到极小值所需的迭代次数明显高于本文。可见，Rand、Logistic、Circle等函数搜索过程随机性较大，致使算法的收敛速度和精度受到显著影响；本文基于最优区间回避采样的搜索策略能较好地保证算法的全局寻优效果，有效提高算法的收敛速度和精度。

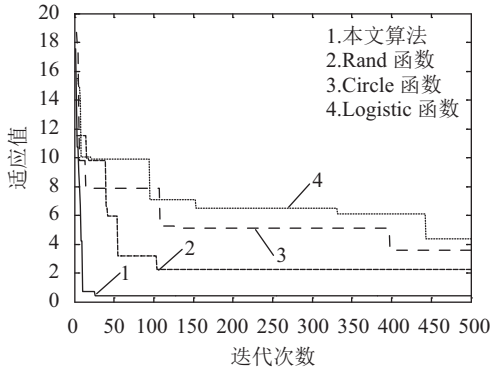

图 3　Rand、Logistic、Circle函数与本文算法比较

Fig.3　Comparison of the functions of Rand, Logistic and circle and the proposed algorithm

3.4　不同算法的收敛精度及稳定性比较

针对每种优化算法，设置函数F_1～F_6的维度D分别取2、4、6、8、10，并分别针对每个D值进行100次实验，最后计算100次实验所得函数的最优值均值μ和最优值标准差ε，结果如表2所示。由表2可知，当处理目标位置分量为负数的函数F_1与F_6时，FOA及IFOA算法对应的μ值较低；当处理其他函数（F_2、F_3、F_4与F_5）时，FOA算法表现最差，IFFO、IFOA、cFOA对应精度与DPFOA近似，但稳定性稍差，说明基于维度分区的果蝇位置更新策略能有效改善寻优结果的稳定性。综合来看，本文算法在处理多峰函数F_4～F_6时具有一定优势，说明本文基于最优区间回避的分区采样策略及不同种群规模动态调整策略在提高算法收敛精度及稳定性、协调算法局部搜索及全局搜索强度方面起到一定作用。

3.5　不同算法的收敛速度比较

函数F_1～F_6的维度D分别取2、4、6、8、10前提下，针对每种参数优化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N_C并将其作为反映算法收敛速度大小的指标，所得结果如表3所示。由表3可知，FOA收敛最快，IFFO与cFOA算法收敛次数近似且普遍小于其他算法，原因在于这两种算法并未在种群收敛后实施全局搜索，因此虽收敛较快但面临过早陷入局部极值的风险。与FOA、IFFO、cFOA等算法相比，本文因全局搜索操

作导致算法对应N_C值偏高；与IFOA与DPFOA算法相比，本文算法对应的N_C值明显偏低，说明算法对应的收敛速度得到显著改善。结合表2、3可知，本文能在提升算法优化精度的同时有效加快算法全局搜索的效率，这主要是因为：1）基于最优区间回避分区采样的搜索果蝇搜索策略能较好地降低全局搜索的盲目性，提高算法的收敛速度；2）不同类型种群规模动态调整策略能灵活调整跟随果蝇种群规模，使得算法在保证全局寻优效果的前提下较好地实现种群的快速收敛。

表 2　不同算法的优化精度及稳定性比较

Tab.2　Comparison of optimization accuracy and stability of different algorithms

函数	FOA算法		IFFO算法		IFOA算法		cFOA算法		DPFOA算法		本文算法	
	μ	ε	μ	ε	μ	ε	μ	ε	μ	ε	μ	ε
F_1	6.62	1.81	3.25×10^{-5}	4.23×10^{-5}	6.13	1.64	6.25×10^{-6}	3.15×10^{-7}	2.56×10^{-6}	3.63×10^{-7}	1.58×10^{-5}	5.55×10^{-10}
F_2	1.28	1.56	8.97×10^{-1}	6.47×10^{-2}	3.81×10^{-2}	5.74×10^{-3}	3.53×10^{-4}	1.28×10^{-3}	2.46×10^{-3}	4.18×10^{-4}	1.19×10^{-4}	2.36×10^{-6}
F_3	5.36×10^{-4}	1.38×10^{-4}	2.56×10^{-4}	3.55×10^{-6}	5.68×10^{-4}	6.37×10^{-5}	3.99×10^{-5}	6.55×10^{-5}	4.13×10^{-4}	5.29×10^{-6}	3.68×10^{-5}	2.55×10^{-7}
F_4	2.39×10^{-1}	5.66×10^{-1}	5.12×10^{-2}	3.68×10^{-3}	2.58×10^{-2}	1.36×10^{-3}	3.38×10^{-3}	8.26×10^{-4}	1.85×10^{-2}	2.08×10^{-4}	6.34×10^{-5}	2.05×10^{-6}
F_5	3.58×10^{-2}	4.12×10^{-2}	3.69×10^{-3}	2.86×10^{-5}	3.66×10^{-4}	1.27×10^{-5}	4.26×10^{-4}	9.02×10^{-5}	1.61×10^{-4}	2.37×10^{-7}	3.59×10^{-5}	2.66×10^{-7}
F_6	6.59	1.26	4.57×10^{-4}	2.03×10^{-6}	6.26	0.97	1.29×10^{-4}	3.96×10^{-6}	6.88×10^{-4}	3.26×10^{-5}	8.25×10^{-6}	1.93×10^{-9}

表 3　不同算法对应的平均迭代次数比较

Tab.3　Comparison of average iterations of different algorithms

函数	FOA算法	IFFO算法	IFOA算法	cFOA算法	DPFOA算法	本文算法
F_1	689	987	1 675	1 156	1 438	1 236
F_2	925	936	1 401	1 071	1 616	1 139
F_3	869	956	1 569	926	1 769	989
F_4	968	1 201	1 533	1 011	1 625	1 136
F_5	830	912	2 236	1 089	2 531	1 256
F_6	975	933	2 463	996	2 274	1 527

3.6　不同算法的综合性能比较

分别针对每个测试函数使用不同优化算法进行100次实验，并按照式（20）～（22）计算不同算法对应的综合性能大小*PEF*，所得结果如图4所示。

$$PEF=\left(\sum_{i=1}^{6}PEF_i\right)/6 \tag{20}$$

$$PEF_i=(E_{iN}+D_{iN}+T_{iN})/3 \tag{21}$$

$$E_{iN}=\frac{E_{iM}-E_i}{E_{iM}-E_{im}},D_{iN}=\frac{D_{iM}-D_i}{D_{iM}-D_{im}},T_{iN}=\frac{T_{iM}-T_i}{T_{iM}-T_{im}} \tag{22}$$

其中，PEF_i为优化算法在在测试函数F_i上的综合性能大小，E_{iN}、D_{iN}、T_{iN}分别为将优化函数对应的最优值均值、标准差均值及收敛次数均值规一化至[0, 1]区间后的结果，E_i、E_{iM}、E_{im}分别代表在函数F_i上不同优化函数对应的最优值均值、最优值均值最大值及最小值，D_i、D_{iM}、D_{im}分别代表算法在F_i上不同优化函数对应的最优值标准差、最优值标准差最大值及最小值，T_i、T_{iM}、T_{im}分别代表算法在F_i上不同优化函数对应的收敛次数均值、收敛次数均值最大值及最小值。可见，在处理较为简单的单峰函数时，本文算法优势

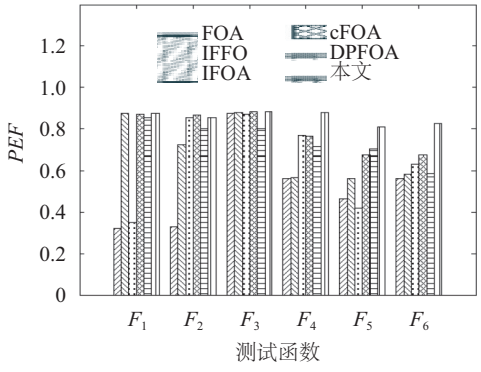

图 4　不同算法的综合性能比较

Fig.4　Comparison of comprehensive performances of different algorithms

表现并不明显；但在处理较为复杂的多峰函数时，本文方法所得*PEF*值普遍多于其他算法，说明本文在保证算法收敛精度的同时有效提高了算法的收敛稳定性和收敛速度，获得更好的综合性能。

3.7　异常检测仿真实验

异常检测将偏离正常用户行为视为入侵嫌疑，该技术可以检测到针对系统的未知攻击，于是成为入侵检测技术研究的重点[14]。目前，针对异常检测主

要采用机器学习方法，包括神经网络（neural networks, NN）、朴素贝叶斯模型（naive bayesian model, NBM）、支持向量机（support vector machine, SVM）等。在基于SVM的异常检测中，惩罚系数c及RBF核函数中的g系数是影响分类性能的重要参数[15]。c值过大或过小，分类器的泛化能力将变差；g值过大或过小，分类准确率将降低。为验证本文算法的参数优化效果，采用LibSVM作为训练和测试工具，并按照下面步骤对标准数据集KDDcup99[16]进行异常检测实验：首先，参考文献[16]对数据集进行数字化、归一化等预处理；接着，为降低计算耗时，采用信息增益方法从数据集41个特征中选择10个特征并用于后续的SVM训练和分类；最后，以分类准确率为目标函数[17]，使用不同算法优化c、g系数。公平起见，设置相关参数如下：c、g最小值$x_{\min}$=0.000 1，c、g最大值$x_{\max}$=5，果蝇数量N=10，最大迭代次数T=200。在此基础上，统计各算法中目标函数值随迭代次数的变化情况及收敛状态下对应的c、g值，结果如图5所示。

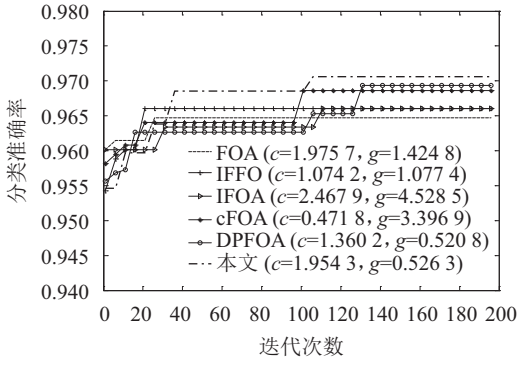

图 5　不同算法对应的目标函数值随迭代次数变化情况

Fig.5　Comparison of objective function values of different algorithms when the iteration number changes

由图5可知，本文达到收敛状态所需的迭代次数约为100次，明显小于DPFOA算法对应的结果（130次），这说明在迭代前期使用以跟随果蝇为主的优化策略能较明显地提高算法收敛速度。进一步发现，本文在收敛状态下对应的分类精度约为0.971，而其他算法对应的分类精度均小于0.970，这说明本文基于分区采样及群密度的果蝇优化算法能较好地避免局部最优，在获取异常检测分类器重要参数的最佳取值方面起到一定作用。

4　结　论

提出了一种基于群密度的改进果蝇优化算法，主要内容包括：1）提出一种基于最优区间回避的分区采样策略更新搜索果蝇的位置，避免全局搜索过程中因盲目产生果蝇新位置而给算法的精度及稳定性带来影响；2）提出一种基于群密度的种群规模动态调整策略，有效协调算法的全局搜索及局部搜索能力。针对6个典型测试函数及异常检测仿真的实验表明，本文算法在收敛精度、收敛速度、稳定性等方面相比传统优化算法具有明显的优势。下一步将研究如何有效提高算法计算效率，并将其扩展到风险评估、工业控制等更多领域。

参考文献：

[1] Liu E,Dong Y,Song J,et al.A modified particle swarm optimization algorithm[J].Natural Science,2015,1(2):151–155.

[2] Li Z T,Liu J.A multi-agent genetic algorithm for community detection in complex networks[J].Physica A:Statistical Mechanics and Its Applications,2016,449:336–347.

[3] Padberg M.Harmony search algorithms for binary optimization problems[M]//Operations Research Proceedings Zurich:Springer,2012:343–348.

[4] Pan Wenchao.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 Sciences Edition),2011,29(4):1–5.[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.]

[5] 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.

[6] Wang X G,Zou Z J.FOA-based SVM parameter optimization and its application in ship manoeuvring prediction[J].Journal of Shanghai Jiaotong University,2013,47(6):884–888.

[7] Han J,Wang P,Yang X.Tuning of PID controller based on fruit fly optimization algorithm[C] //International Conference on Mechatronics and Automation.Chengdu:IEEE,2012:409–413.

[8] Li H Z,Guo S,Li C J,et al.A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm[J].Knowledge-Based Systems,2013,37(2):378–387.

[9] 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.

[10] Wang Youwei,Zhu Jianming,Feng Lizhou,et al.Novel fly optimization algorithm based on near suburb and far suburb[J].Computer Engineering,2017,43(2):210–214.[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.]

[11] Wang L,Shi Y L,Liu S.An improved fruit fly optimization algorithm and its application to joint replenishment problems[J].Expert Systems with Applications,2015,42(9):4310–4323.

[12] Mitic M,Vukovic N,Petrovic M,et al.Chaotic fruit fly optimization algorithm[J].Knowledge-Based Systems,2015,89(C):446–458.

[13] Wang Youwei,Feng Lizhou,Zhu Jianming.Novel fruit fly optimization algorithm based on dimension partition[J].Computer Science,2016,43(12):264–268.[王友卫,凤丽洲,朱建明.基于维度分区的果蝇优化新算法[J].计算机科学,2016,43(12):264–268.]

[14] Savage D,Zhang X,Yu X,et al.Anomaly detection in online social networks[J].Social Networks,2016,39(1):62–70.

[15] Chang C C,Lin C J.LIBSVM:A library for support vector machines[J].Acm Transactions on Intelligent Systems & Technology,2007,2(3):27.

[16] Wu Xiaonian,Peng Xiaojin,Yang Yuyang,et al.Two-level feature selection method based on SVM for intrusion detection[J].Journal on Communications,2015,36(4):19–26.[武小年,彭小金,杨宇洋,等.入侵检测中基于SVM的两级特征选择方法[J].通信学报,2015,36(4):19–26.]

[17] Yang J,Liu Y,Zhu X,et al.A new feature selection based on comprehensive measurement both in inter-category and intra-category for text classification[J].Information Processing Manage,2012,48(4):741–754.

（编辑　李轶楠）

引用格式:Wang Youwei,Zhu Jianming,Feng Lizhou.Improved fruit fly optimization algorithm based on population density and its application in anomaly detection[J].Advanced Engineering Sciences,2017,49(5):127–134.[王友卫, 朱建明, 凤丽洲. 基于群密度的改进果蝇优化算法及在异常检测中的应用[J].工程科学与技术,2017,49(5):127–134.]