**蚁狮算法综述**

1821011733 陈俊文

**摘要** 蚁狮算法是Seyedali Mirjalili 在2015年提出的一种基于蚁狮觅食行为的群体智能算法。因其强大的性能深受研究者的青睐，得以广泛的研究和应用。本文首先简要介绍了群体智能，然后详细介绍了蚁狮算法的原理及实现步骤，最后综述近年来国内外对该算法及其应用的研究状况。

**关键字** 蚁狮算法；群体智能；觅食行为

**A Compreheive Survey On Ant Lion** **Algorithm**

1821011733 Chen Junwen

**Abstract** Ant lion algorithm is a kind of swarm intelligence algorithm based on ant lion foraging behavior which is proposed by Seyedali Mirjalili on 2015. The algorithm get many researchers’ favor because of its powerful performance and be researched and used widely. Firstly, this paper introduces the development of swarm intelligence is briefly. Secondly, this paper introduces the principle and steps of ant lion algorithm in detail. Finally, the research status of the algorithm and its application in recent years is reviewed.

**Keyword** Ant Lion Algorithm; Swarm Intelligence; Foraging Behavior

**1 引言**

群智能算法都是基于生物群体行为产生的。生物群中每个个体只有简单的信息处理能力和行为能力。例如，鸟群拥有飞行、捕食和避免碰撞的能力；昆虫拥有爬行、觅食以及产生信息素的能力。群体中各个个体之间可以进行信息交互。鸟群可以进行视觉，听觉和磁场的信息交互；昆虫可以感知彼此的信息素。群体的能力要远远大于个体能力的简单叠加。群体拥有信息感知能力，分工协作能力和适应生存能力。例如，鸟群协作捕食，鱼群合作逃避捕食者。这些行为都是群体才能完成的行为。

群智能概念主要由五条基本原则体现：

（1）群内个体具有能执行简单的时间或空间上的评估和计算的能力。

（2）群内个体能对环境（包括群内其他个体）的关键性因素的变化做出的响应。

（3）群内不同个体对环境中的某一变化所表现出的响应行为具有多样性。

（4）不是每次环境的变化都会导致整个群体的行为模式的改变。

（5）环境所发生的变化中，若出现群体值得付出代价的改变机遇，群体必须能够改变其行为模式。

广义的群智能算法主要有粒子群算法，模拟鸟群觅食行为[1]；蚁群算法，模拟蚁群觅食行为[2]；萤火虫算法，模拟萤火虫闪烁的行为[3]；蜂群算法，模拟蜜蜂采蜜行为[4]；以及混合算法，是多种群智能算法的结合[5]。

群智能算法可以应用于多个方面。在优化领域，群智能算法可以用于函数优化[6]，路径优化，组合优化问题（0-1背包），多目标优化以及高维复杂优化。在调度领域[7]，群智能算法可以用于物流云服务调度，流水线调度。在网络安全领域[8]，群智能算法可以用于密码学，例如密钥生成、S盒设计、数字水印；以及网络入侵，例如攻击源定位、制定入侵分类规则、聚类分析。在识别决策领域[9]，群智能算法可以用于医学成像分析，病症诊断和图像分割。

**2 蚁狮算法**

蚁狮是一种狡诈而凶残的昆虫，生活在气候炎热地区的干沙里。其捕食方法颇为巧妙。蚁狮优化方法是一种自然启发算法，遵循蚁狮幼虫的狩猎行为[10]。蚁狮幼虫通过沿着沙子中的圆形路径移动并用巨大的下颚投掷沙子来创建一个圆锥形孔。在挖掘陷阱后，幼虫躲在锥体的底部，等待蚂蚁被困在坑中。一旦蚁狮意识到猎物陷入陷阱，蚁狮将沙子向外抛出并将猎物滑入坑中。当猎物被捕入时，蚂蚁将猎物拉向自身并消灭掉。这种方法有五个主要步骤：首先蚂蚁进行随机行走，然后蚁狮构建陷阱，当蚂蚁掉落陷阱时，蚁狮开始捕获猎物，最后蚁狮重建陷阱。

蚂蚁使用随机游走在搜索空间中移动，这些搜索空间受到了敌人陷阱的影响。在每次迭代时随机步行更新蚂蚁的位置。迭代t的随机游走是使用等式（1）。但是，为了确保所有随机游走都落在搜索空间的边界内，应用标准化。 使用等式（2）将随机游走归一化。

X[t]=[0,cumsum(2r()-1), cumsum(2r()-1),…cumsum(2r()-1)] (1)

其中，r(t)=1如果rand＞0.5或r(t)=0如果rand≤0.5

=+ (2)

和分别是变量的随机游走的最小值和最大值。是迭代t时第i个变量的最小值。是迭代t处的第i个变量的最大值。

蚁狮造成的陷阱会影响蚂蚁的随机行走。使用方程式（3）和（4）在数学上解释该过程。

=+ (3)

=+ (4)

在这里，是蚁狮 i在迭代t的位置。和分别是所有变量的最小值和最大值。是迭代t时第i 个蚂蚁的最小值，是迭代t时第i 个蚂蚁的最大值。

轮盘赌选择方法用于基于其适合度值选择用于优化的蚁狮。在每次迭代中获得的最适合的蚁狮被保存为精英。精英影响蚂蚁的运动。此外，蚂蚁的位置根据所选的蚁狮以及精英的随机游走进行更新，因为每只蚂蚁都会围绕选定的蚁群和精英周围行走。这个过程用公式（5）表示。

= (5)

是在第t次迭代时由轮盘赌轮选择的围绕蚁狮的随机游走，是在第t次迭代时围绕精英的随机游走。是迭代t时蚂蚁i的位置。

计算所有蚂蚁的适合度值。 如果蚂蚁比蚁狮具有更好的适应性，则用相应的蚂蚁代替蚁狮。 同样地，如果蚁狮比精英有更好的适应性，精英也会被一个蚁狮取代。

蚁狮算法的流程图如图1所示。



图1 蚁狮算法流程图

**3 算法应用的研究**

蚁狮算法是一种随机全局优化算法，不仅具有探索搜索空间的高潜力，而且具有高度开发能力，可以快速收敛以实现全局最优化。

**3.1 蚁狮算法与****K-means算法混合进行最优聚类分析**

Santosh Kumar Majhi和Shubhra Biswal在一篇2018年的英文论文中提出将蚁狮优化算法和K-means算法结合[11]。K-Means是典型的聚类算法，它要求输入簇的数目k以及包含n个对象的数据集D，输出k个簇的集合。K-Means聚类的过程如下：

1) 从数据集D中任意选择k个对象作为初始簇中心

2) 开始迭代

3) 根据簇中对象的均值，将每个对象分配到最相似的簇

4) 更新簇均值，即重新计算每个簇中对象的均值

5) 直到聚类结果不再发生变化（本轮形成的簇与前一轮形成的簇相同）

混合K-Means和蚁狮聚类算法的流程图如图2所示，首先执行K-Means算法，接着执行蚁狮算法，最后还要执行K-Means算法。

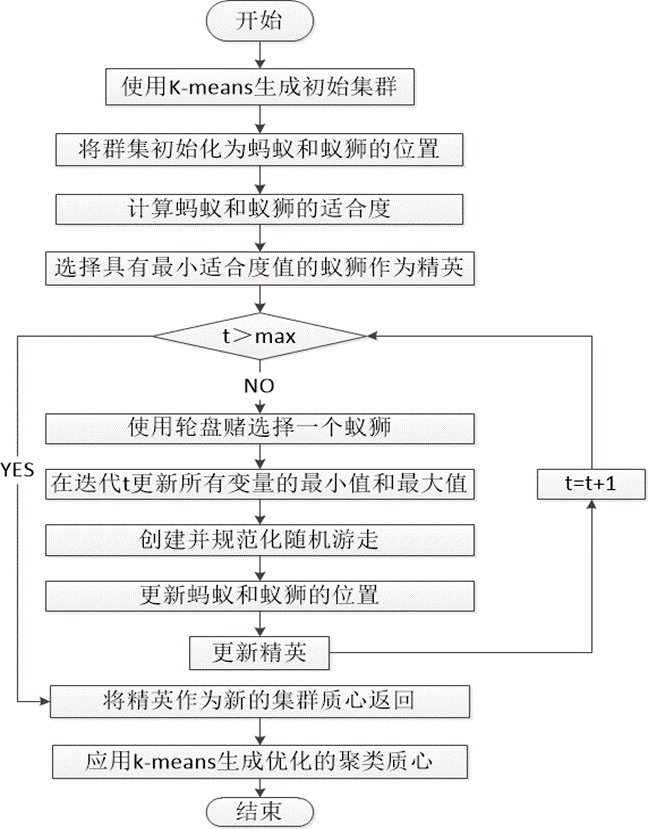


图2 K-Means和蚁狮算法的混合聚类算法流程图

聚类算法通常采用UCI机器学习库中的数据集[12]来分析所提算法的有效性。表1给出了一些常用的数据集。

表1 数据集

|  |  |  |  |
| --- | --- | --- | --- |
| 数据集 | 簇数 | 特征值 | 描述 |
| 鸢尾花数据集 | 3 | 4 | 150（50,50,50） |
| 葡萄酒数据集 | 3 | 13 | 178（59.71.48） |
| 玻璃数据集 | 6 | 9 | 214（70,76,17,13,9,29） |
| 威斯康星州乳腺癌数据集 | 2 | 9 | 683（444,239） |
| 元音数据集 | 6 | 3 | 871（72,89,172,151,207,180） |
| Hill Valley数据集 | 2 | 101 | 606（305,301） |
| CMC数据集 | 3 | 9 | 1473（629,334,510） |

算法性能主要通过群内距离总和的平均值和F-度量来评估。

群内距离有三种方法描述：质心直径方法中群内距离是群集质心与属于该群集的所有点之间的平均距离；完全直径法通过计算属于群集的任意两点之间的最大距离来计算群内距离；平均直径方法考虑属于群集的所有点对之间的平均距离。

F-度量[13]则由精度和召回率来计算。一个对象的精度P是指同一簇中有多少其他对象与该对象属于同一个类别，是针对预测结果而言的。而一个对象的召回率R是指有多少同一类别的对象被分配在相同的簇中。，是针对原样本而言的。F-度量由式（6）给出：

F1= (6)

为了获得更好的簇质量，群内距离的值应该是最小的，并且F-度量的值应该是最大的。以K-Means和蚁狮算法的混合聚类算法为例，比较K-Means，KMeans-PSO，KMeans-FA（萤火虫算法），基于密度的噪声应用空间聚类DBSCAN，修订的DBSCAN和KMeans-ALO算法（K-Means和蚁狮算法的混合聚类算法），在100,500和1000次迭代的玻璃数据集上的10次不同运行获得的结果。如表2可知，KMeans-ALO，KMeans-FA和KMeans-PSO，对于100次迭代的集群内距离的平均值的总和给出相同的最小值0.2663。对于500次迭代，KMeans-PSO和KMeans-FA都给出最小的群内距离，即0.2663。对于1000次迭代，KMeans-PSO和K-Means给出最小的群内距离。在100次迭代时，K-Means，KMeans-ALO，KMeans-FA和KMeans-PSO给出了最大F-度量的值0.9989。在500次和1000次迭代中，K-Means给出了最大F-度量的值1.4320。

**3.2 蚁狮算法其他应用**

蚁狮算法除了优化KMeans算法，还有其他应用。黄长强，赵克新在蚂蚁的行为中引入混沌调节因子,在蚁狮的行为中引入反调节因子,提高了算法的探索能力和开发能力;其次在建立3维环境模型的基础上,充分利用地形和约束信息,缩减搜索空间;最后将改进后的算法应用于3维航迹规划[14]。景坤雷等提出一种具有Levy变异和精英自适应竞争机制的蚁狮优化算法应用于硅单晶热场温度模型的参数辨识[15]。 赵世杰等在蚁狮原有信息的基础上引进混沌搜索机制，提出了一种带混沌侦查机制的蚁狮优化算法用于支持向量机参数的优化[16]。李宗妮等提出一种采用改进蚁狮优化算法(RBALO)的图像增强方法.应用于图像增强[17]。

表2性能指标比较

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 算法 | 迭代 | 最好值 | 平均值 | 最差值 | F-度量 | 标准差 |
| K-means | 100 | 0.2693 | 0.27678 | 0.3053 | 0.9989 | 0.0174 |
| KMeans-PSO | 100 | 0.2663 | 0.28374 | 0.3055 | 0.9989 | 0.0186 |
| K-means-FA | 100 | 0.2663 | 0.28758 | 0.3051 | 0.9989 | 0.0184 |
| DBSCAN | 100 | 0.2681 | 0.2681 | 0.2681 | 0.0280 | 0 |
| 修订DBSCAN | 100 | 0.2669 | 0.2669 | 0.2669 | 0.8928 | 0 |
| KMeans-ALO | 100 | 0.2663 | 0.28287 | 0.3051 | 0.9989 | 0.0192 |
| K-means  KMeans-PSO  Kmeans-FA  DBSCAN  修订DBSCAN  KMeans-ALO | 500  500  500  500  500  500 | 0.2665  0.2663  0.2663  0.2681  0.2669  0.2664 | 0.28674  0.291  0.28078  0.2681  0.2669  0.29027 | 0.3053  0.3053  0.3051  0.2681  0.2669  0.3051 | 1.4320  0.9989  1.4320  0.0280  0.8928  0.9989 | 0.0195  0.0184  0.0169  0  0  0.0192 |
| K-Means  KMeans-PSO  KMeans-FA  DBSCAN  修订DBSCAN  KMeans-ALO | 1000  1000  1000  1000  1000  1000 | 0.2665  0.2664  0.2697  0.2681  0.2669  0.2664 | 0.2865  0.27998  0.29165  0.2681  0.2669  0.29027 | 0.3057  0.3051  0.3051  0.2681  0.2669  0.3051 | 1.4320  0.9989  0.9989  0.0280  0.8928  0.9989 | 0.0199  0.0173  0.0173  0  0  0.0192 |

**4 总结**

蚁狮算法是一种群智能优化算法。群智能算法都是基于生物群体行为产生的，群体的力量要远远大于个体。蚁狮算法是一种随机全局优化算法，是模拟蚁狮幼虫的狩猎行为。蚁狮优化算法不仅用于优化KMeans算法，还可经过变形用于3维航迹规划，参数辨识等。总而言之，蚁狮算法是一个可用性强的群智能算法。

**参考文献**

[1]杨维,李歧强.粒子群优化算法综述[J].中国工程科学,2004(05):87-94.

[2]陈崚,沈洁,秦玲.蚁群算法求解连续空间优化问题的一种方法[J].软件学报,2002(12):2317-2323.

[3]刘长平,叶春明.一种新颖的仿生群智能优化算法:萤火虫算法[J].计算机应用研究,2011,28(09):3295-3297.

[4]秦全德,程适,李丽,史玉回.人工蜂群算法研究综述[J].智能系统学报,2014,9(02):127-135.

[5]邹挺. 基于蚁群和人工鱼群混合群智能算法在物流配送路径优化问题中的应用研究[D].苏州大学,2011.

[6]唐超礼. 群智能算法及其在函数优化中的应用研究[D].安徽理工大学,2007.

[7]唐海波,叶春明.仿生群智能算法在生产调度中的应用综述[J].工业工程,2010,13(03):1-5+42.

[8]杨义先,李丽香,彭海朋,袁静,陈永刚,张浩.群体智能算法及其在信息安全中的应用探索[J].信息安全学报,2016,1(01):39-49.

[9]杨振宇. 基于群智能算法的图像分割方法研究[D].电子科技大学,2010.

[10] S. Mirjalili, The ant lion optimizer, Adv. Eng. Software 83(2015) 80e98.

[11]Santosh Kumar Majhi,Shubhra Biswal. Optimal cluster analysis using hybrid K-Means and Ant Lion Optimizer[J]. Karbala International Journal of Modern Science,2018.

[12]<http://archive.ics.uci.edu/ml/datasets.html>

[13]https://www.cnblogs.com/bluepoint2009/archive/2012/09/18/precision-recall-f\_measures.html

[14]黄长强,赵克新.基于改进蚁狮算法的无人机三维航迹规划[J].电子与信息学报,2018,40(7):1532-1538.DOI:10.11999/JEIT170961.

[15]景坤雷, 赵小国, 张新雨, et al. 具有Levy变异和精英自适应竞争机制的蚁狮优化算法[J]. 智能系统学报, 2018, 13(02):236-242.

[16]赵世杰, 高雷阜, 于冬梅, et al. 带混沌侦查机制的蚁狮优化算法优化SVM参数[J]. 计算机科学与探索, 2016, 10(5):722-731.

[17] 李宗妮, 吴伟民, 林志毅. 一种采用改进蚁狮优化算法的图像增强方法[J]. 计算机应用研究, 2018, 35(4): 1258-1260.

**附录 蚁狮算法MATLAB代码**

**ALO.m**

function [Elite\_antlion\_fitness,Elite\_antlion\_position,Convergence\_curve]=ALO(N,Max\_iter,lb,ub,dim,fobj)

% 初始化蚁狮和蚂蚁的位置

antlion\_position=initialization(N,dim,ub,lb);

ant\_position=initialization(N,dim,ub,lb);

% 初始化变量以保存精英的位置，排序的蚁群，收敛曲线，蚁群适应度和蚂蚁适应度Sorted\_antlions=zeros(N,dim);

Elite\_antlion\_position=zeros(1,dim);

Elite\_antlion\_fitness=inf;

Convergence\_curve=zeros(1,Max\_iter);

antlions\_fitness=zeros(1,N);

ants\_fitness=zeros(1,N);

%计算初始蚁群的适应度并对它们进行排序

for i=1:size(antlion\_position,1)

antlions\_fitness(1,i)=fobj(antlion\_position(i,:));

end

[sorted\_antlion\_fitness,sorted\_indexes]=sort(antlions\_fitness);

for newindex=1:N

Sorted\_antlions(newindex,:)=antlion\_position(sorted\_indexes(newindex),:);

end

Elite\_antlion\_position=Sorted\_antlions(1,:);

Elite\_antlion\_fitness=sorted\_antlion\_fitness(1);

% 主循环从第二次迭代开始，因为第一次迭代专用于计算蚁狮的适应度

Current\_iter=2;

while Current\_iter<Max\_iter+1

% 这个for循环模拟随机游走

for i=1:size(ant\_position,1)

% 根据他们的适应度选择蚁狮（更好的蚁狮捕捉蚂蚁的机会更高）

Rolette\_index=RouletteWheelSelection(1./sorted\_antlion\_fitness);

if Rolette\_index==-1

Rolette\_index=1;

end

% RA是由轮盘赌轮随机围绕选定的蚁狮 RA=Random\_walk\_around\_antlion(dim,Max\_iter,lb,ub, Sorted\_antlions(Rolette\_index,:),Current\_iter);

% RA是精英周围的随机游走（迄今为止最好的蚁狮）

[RE]=Random\_walk\_around\_antlion(dim,Max\_iter,lb,ub, Elite\_antlion\_position(1,:),Current\_iter);

ant\_position(i,:)= (RA(Current\_iter,:)+RE(Current\_iter,:))/2;

end

for i=1:size(ant\_position,1)

% 边界检查（如果超出边界，则将搜索空间内的蚂蚁带回来）

Flag4ub=ant\_position(i,:)>ub;

Flag4lb=ant\_position(i,:)<lb;

ant\_position(i,:)=(ant\_position(i,:).\*(~(Flag4ub+Flag4lb)))+ub.\*Flag4ub+lb.\*Flag4lb;

ants\_fitness(1,i)=fobj(ant\_position(i,:));

end

% 更新基于蚂蚁的蚁狮位置和适合度（如果蚂蚁变得比蚁狮更健康，我们认为它是由蚁狮抓捕而且蚁巢更新到了构建陷阱的位置）

double\_population=[Sorted\_antlions;ant\_position];

double\_fitness=[sorted\_antlion\_fitness ants\_fitness];

[double\_fitness\_sorted I]=sort(double\_fitness);

double\_sorted\_population=double\_population(I,:);

antlions\_fitness=double\_fitness\_sorted(1:N);

Sorted\_antlions=double\_sorted\_population(1:N,:);

% 如果任何蚂蚁变得比它更适应，则更新精英的位置

if antlions\_fitness(1)<Elite\_antlion\_fitness

Elite\_antlion\_position=Sorted\_antlions(1,:);

Elite\_antlion\_fitness=antlions\_fitness(1);

end

% Keep the elite in the population

Sorted\_antlions(1,:)=Elite\_antlion\_position;

antlions\_fitness(1)=Elite\_antlion\_fitness;

% 更新收敛曲线 Convergence\_curve(Current\_iter)=Elite\_antlion\_fitness;

% 显示到目前为止获得的迭代和最佳值

if mod(Current\_iter,50)==0

display(['At iteration ', num2str(Current\_iter), ' the elite fitness is ', num2str(Elite\_antlion\_fitness)]);

end

Current\_iter=Current\_iter+1;

end

**func\_plot.m**

% 此函数绘制基准函数

function func\_plot(func\_name)

[lb,ub,dim,fobj]=Get\_Functions\_details(func\_name);

switch func\_name

case 'F1'

x=-100:2:100; y=x; %[-100,100]

case 'F2'

x=-100:2:100; y=x; %[-10,10]

case 'F3'

x=-100:2:100; y=x; %[-100,100]

case 'F4'

x=-100:2:100; y=x; %[-100,100]

case 'F5'

x=-200:2:200; y=x; %[-5,5]

case 'F6'

x=-100:2:100; y=x; %[-100,100]

case 'F7'

x=-1:0.03:1; y=x %[-1,1]

case 'F8'

x=-500:10:500;y=x; %[-500,500]

case 'F9'

x=-5:0.1:5; y=x; %[-5,5]

case 'F10'

x=-20:0.5:20; y=x;%[-500,500]

case 'F11'

x=-500:10:500; y=x;%[-0.5,0.5]

case 'F12'

x=-10:0.1:10; y=x;%[-pi,pi]

case 'F13'

x=-5:0.08:5; y=x;%[-3,1]

case 'F14'

x=-100:2:100; y=x;%[-100,100]

case 'F15'

x=-5:0.1:5; y=x;%[-5,5]

case 'F16'

x=-1:0.01:1; y=x;%[-5,5]

case 'F17'

x=-5:0.1:5; y=x;%[-5,5]

case 'F18'

x=-5:0.06:5; y=x;%[-5,5]

case 'F19'

x=-5:0.1:5; y=x;%[-5,5]

case 'F20'

x=-5:0.1:5; y=x;%[-5,5]

case 'F21'

x=-5:0.1:5; y=x;%[-5,5]

case 'F22'

x=-5:0.1:5; y=x;%[-5,5]

case 'F23'

x=-5:0.1:5; y=x;%[-5,5]

end

L=length(x);

f=[];

for i=1:L

for j=1:L

if strcmp(func\_name,'F15')==0 && strcmp(func\_name,'F19')==0 && strcmp(func\_name,'F20')==0 && strcmp(func\_name,'F21')==0 && strcmp(func\_name,'F22')==0 && strcmp(func\_name,'F23')==0

f(i,j)=fobj([x(i),y(j)]);

end

if strcmp(func\_name,'F15')==1

f(i,j)=fobj([x(i),y(j),0,0]);

end

if strcmp(func\_name,'F19')==1

f(i,j)=fobj([x(i),y(j),0]);

end

if strcmp(func\_name,'F20')==1

f(i,j)=fobj([x(i),y(j),0,0,0,0]);

end

if strcmp(func\_name,'F21')==1 || strcmp(func\_name,'F22')==1 ||strcmp(func\_name,'F23')==1

f(i,j)=fobj([x(i),y(j),0,0]);

end

end

end

surfc(x,y,f,'LineStyle','none');

end

**Get\_Functions\_details.m**

function [lb,ub,dim,fobj] = Get\_Functions\_details(F)

switch F

case 'F1'

fobj = @F1;

lb=-100;

ub=100;

dim=10;

case 'F2'

fobj = @F2;

lb=-10;

ub=10;

dim=10;

case 'F3'

fobj = @F3;

lb=-100;

ub=100;

dim=10;

case 'F4'

fobj = @F4;

lb=-100;

ub=100;

dim=10;

case 'F5'

fobj = @F5;

lb=-30;

ub=30;

dim=10;

case 'F6'

fobj = @F6;

lb=-100;

ub=100;

dim=10;

case 'F7'

fobj = @F7;

lb=-1.28;

ub=1.28;

dim=10;

case 'F8'

fobj = @F8;

lb=-500;

ub=500;

dim=10;

case 'F9'

fobj = @F9;

lb=-5.12;

ub=5.12;

dim=10;

case 'F10'

fobj = @F10;

lb=-32;

ub=32;

dim=10;

case 'F11'

fobj = @F11;

lb=-600;

ub=600;

dim=10;

case 'F12'

fobj = @F12;

lb=-50;

ub=50;

dim=10;

case 'F13'

fobj = @F13;

lb=-50;

ub=50;

dim=10;

case 'F14'

fobj = @F14;

lb=-65.536;

ub=65.536;

dim=2;

case 'F15'

fobj = @F15;

lb=-5;

ub=5;

dim=4;

case 'F16'

fobj = @F16;

lb=-5;

ub=5;

dim=2;

case 'F17'

fobj = @F17;

lb=[-5,0];

ub=[10,15];

dim=2;

case 'F18'

fobj = @F18;

lb=-2;

ub=2;

dim=2;

case 'F19'

fobj = @F19;

lb=0;

ub=1;

dim=3;

case 'F20'

fobj = @F20;

lb=0;

ub=1;

dim=6;

case 'F21'

fobj = @F21;

lb=0;

ub=10;

dim=4;

case 'F22'

fobj = @F22;

lb=0;

ub=10;

dim=4;

case 'F23'

fobj = @F23;

lb=0;

ub=10;

dim=4;

end

end

% F1

function o = F1(x)

o=sum(x.^2);

end

% F2

function o = F2(x)

o=sum(abs(x))+prod(abs(x));

end

% F3

function o = F3(x)

dim=size(x,2);

o=0;

for i=1:dim

o=o+sum(x(1:i))^2;

end

end

% F4

function o = F4(x)

o=max(abs(x));

end

% F5

function o = F5(x)

dim=size(x,2);

o=sum(100\*(x(2:dim)-(x(1:dim-1).^2)).^2+(x(1:dim-1)-1).^2);

end

% F6

function o = F6(x)

o=sum(abs((x+.5)).^2);

end

% F7

function o = F7(x)

dim=size(x,2);

o=sum([1:dim].\*(x.^4))+rand;

end

% F8

function o = F8(x)

o=sum(-x.\*sin(sqrt(abs(x))));

end

% F9

function o = F9(x)

dim=size(x,2);

o=sum(x.^2-10\*cos(2\*pi.\*x))+10\*dim;

end

% F10

function o = F10(x)

dim=size(x,2);

o=-20\*exp(-.2\*sqrt(sum(x.^2)/dim))-exp(sum(cos(2\*pi.\*x))/dim)+20+exp(1);

end

% F11

function o = F11(x)

dim=size(x,2);

o=sum(x.^2)/4000-prod(cos(x./sqrt([1:dim])))+1;

end

% F12

function o = F12(x)

dim=size(x,2);

o=(pi/dim)\*(10\*((sin(pi\*(1+(x(1)+1)/4)))^2)+sum((((x(1:dim-1)+1)./4).^2).\*...

(1+10.\*((sin(pi.\*(1+(x(2:dim)+1)./4)))).^2))+((x(dim)+1)/4)^2)+sum(Ufun(x,10,100,4));

end

% F13

function o = F13(x)

dim=size(x,2);

o=.1\*((sin(3\*pi\*x(1)))^2+sum((x(1:dim-1)-1).^2.\*(1+(sin(3.\*pi.\*x(2:dim))).^2))+...

((x(dim)-1)^2)\*(1+(sin(2\*pi\*x(dim)))^2))+sum(Ufun(x,5,100,4));

end

% F14

function o = F14(x)

aS=[-32 -16 0 16 32 -32 -16 0 16 32 -32 -16 0 16 32 -32 -16 0 16 32 -32 -16 0 16 32;,...

-32 -32 -32 -32 -32 -16 -16 -16 -16 -16 0 0 0 0 0 16 16 16 16 16 32 32 32 32 32];

for j=1:25

bS(j)=sum((x'-aS(:,j)).^6);

end

o=(1/500+sum(1./([1:25]+bS))).^(-1);

end

% F15

function o = F15(x)

aK=[.1957 .1947 .1735 .16 .0844 .0627 .0456 .0342 .0323 .0235 .0246];

bK=[.25 .5 1 2 4 6 8 10 12 14 16];bK=1./bK;

o=sum((aK-((x(1).\*(bK.^2+x(2).\*bK))./(bK.^2+x(3).\*bK+x(4)))).^2);

end

% F16

function o = F16(x)

o=4\*(x(1)^2)-2.1\*(x(1)^4)+(x(1)^6)/3+x(1)\*x(2)-4\*(x(2)^2)+4\*(x(2)^4);

end

% F17

function o = F17(x)

o=(x(2)-(x(1)^2)\*5.1/(4\*(pi^2))+5/pi\*x(1)-6)^2+10\*(1-1/(8\*pi))\*cos(x(1))+10;

end

% F18

function o = F18(x)

o=(1+(x(1)+x(2)+1)^2\*(19-14\*x(1)+3\*(x(1)^2)-14\*x(2)+6\*x(1)\*x(2)+3\*x(2)^2))\*...

(30+(2\*x(1)-3\*x(2))^2\*(18-32\*x(1)+12\*(x(1)^2)+48\*x(2)-36\*x(1)\*x(2)+27\*(x(2)^2)));

end

% F19

function o = F19(x)

aH=[3 10 30;.1 10 35;3 10 30;.1 10 35];cH=[1 1.2 3 3.2];

pH=[.3689 .117 .2673;.4699 .4387 .747;.1091 .8732 .5547;.03815 .5743 .8828];

o=0;

for i=1:4

o=o-cH(i)\*exp(-(sum(aH(i,:).\*((x-pH(i,:)).^2))));

end

end

% F20

function o = F20(x)

aH=[10 3 17 3.5 1.7 8;.05 10 17 .1 8 14;3 3.5 1.7 10 17 8;17 8 .05 10 .1 14];

cH=[1 1.2 3 3.2];

pH=[.1312 .1696 .5569 .0124 .8283 .5886;.2329 .4135 .8307 .3736 .1004 .9991;...

.2348 .1415 .3522 .2883 .3047 .6650;.4047 .8828 .8732 .5743 .1091 .0381];

o=0;

for i=1:4

o=o-cH(i)\*exp(-(sum(aH(i,:).\*((x-pH(i,:)).^2))));

end

end

% F21

function o = F21(x)

aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2 9;5 5 3 3;8 1 8 1;6 2 6 2;7 3.6 7 3.6];

cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];

o=0;

for i=1:5

o=o-((x-aSH(i,:))\*(x-aSH(i,:))'+cSH(i))^(-1);

end

end

% F22

function o = F22(x)

aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2 9;5 5 3 3;8 1 8 1;6 2 6 2;7 3.6 7 3.6];

cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];

o=0;

for i=1:7

o=o-((x-aSH(i,:))\*(x-aSH(i,:))'+cSH(i))^(-1);

end

end

% F23

function o = F23(x)

aSH=[4 4 4 4;1 1 1 1;8 8 8 8;6 6 6 6;3 7 3 7;2 9 2 9;5 5 3 3;8 1 8 1;6 2 6 2;7 3.6 7 3.6];

cSH=[.1 .2 .2 .4 .4 .6 .3 .7 .5 .5];

o=0;

for i=1:10

o=o-((x-aSH(i,:))\*(x-aSH(i,:))'+cSH(i))^(-1);

end

end

function o=Ufun(x,a,k,m)

o=k.\*((x-a).^m).\*(x>a)+k.\*((-x-a).^m).\*(x<(-a));

end

**initialization.m**

% 此函数创建第一个随机填充

function X=initialization(SearchAgents\_no,dim,ub,lb)

Boundary\_no= size(ub,2); % numnber of boundaries

% 如果所有变量的边界等于用户输入单个

% number for both ub and lb

if Boundary\_no==1

X=rand(SearchAgents\_no,dim).\*(ub-lb)+lb;

end

% 如果每个变量都有不同的lb和ub

if Boundary\_no>1

for i=1:dim

ub\_i=ub(i);

lb\_i=lb(i);

X(:,i)=rand(SearchAgents\_no,1).\*(ub\_i-lb\_i)+lb\_i;

end

end

**Random\_walk\_around\_antlion.m**

% 此功能可创建随机游走

function [RWs]=Random\_walk\_around\_antlion(Dim,max\_iter,lb, ub,antlion,current\_iter)

if size(lb,1) ==1 && size(lb,2)==1 %检查边界是否是标量 lb=ones(1,Dim)\*lb;

ub=ones(1,Dim)\*ub;

end

if size(lb,1) > size(lb,2) %检查边界向量是水平还是垂直

lb=lb';

ub=ub';

end

I=1;

if current\_iter>max\_iter/10

I=1+100\*(current\_iter/max\_iter);

end

if current\_iter>max\_iter/2

I=1+1000\*(current\_iter/max\_iter);

end

if current\_iter>max\_iter\*(3/4)

I=1+10000\*(current\_iter/max\_iter);

end

if current\_iter>max\_iter\*(0.9)

I=1+100000\*(current\_iter/max\_iter);

end

if current\_iter>max\_iter\*(0.95)

I=1+1000000\*(current\_iter/max\_iter);

end

% Dicrease boundaries to converge towards antlion

lb=lb/(I);

ub=ub/(I);

% Move the interval of [lb ub] around the antlion [lb+anlion ub+antlion]

if rand<0.5

lb=lb+antlion;

else

lb=-lb+antlion;

end

if rand>=0.5

ub=ub+antlion;

else

ub=-ub+antlion;

end

% 此函数创建n个向量的随机游走并对lb和ub进行标准化

for i=1:Dim

X = [0 cumsum(2\*(rand(max\_iter,1)>0.5)-1)'];

%[a b]--->[c d]

a=min(X);

b=max(X);

c=lb(i);

d=ub(i);

X\_norm=((X-a).\*(d-c))./(b-a)+c;

RWs(:,i)=X\_norm;

end

**RouletteWheelSelection.m**

function choice = RouletteWheelSelection(weights)

accumulation = cumsum(weights);

p = rand() \* accumulation(end);

chosen\_index = -1;

for index = 1 : length(accumulation)

if (accumulation(index) > p)

chosen\_index = index;

break;

end

end

choice = chosen\_index;

**main.m**

clear all

clc

SearchAgents\_no=10; % 搜索代理的数量

Function\_name='F11'; % Name of the test function that can be from F1 to F23

Max\_iteration=50; % 最大迭代次数

% 加载所选基准测试功能的详细信息[lb,ub,dim,fobj]=Get\_Functions\_details(Function\_name);

[Best\_score,Best\_pos,cg\_curve]=ALO(SearchAgents\_no,Max\_iteration,lb,ub,dim,fobj);

figure('Position',[500 500 660 290])

%Draw search space

subplot(1,2,1);

func\_plot(Function\_name);

title('Test function')

xlabel('x\_1');

ylabel('x\_2');

zlabel([Function\_name,'( x\_1 , x\_2 )'])

grid off

%Draw objective space

subplot(1,2,2);

semilogy(cg\_curve,'Color','r')

title('Convergence curve')

xlabel('Iteration');

ylabel('Best score obtained so far');

axis tight

grid off

box on

legend('ALO')

display(['The best solution obtained by ALO is : ', num2str(Best\_pos)]);

display(['The best optimal value of the objective funciton found by ALO is : ', num2str(Best\_score)]);