Guide of M

内容概要： 数学建模算法

创建时间： 2022/4/8 12:41 **更新时间：** 2022/4/20 14:00

作者： **TwinkelStar**

ACO 蚁群算法

ACO Algorithm

1、操作系统相关环境

1）硬件环境：

- 电脑

2）软件环境：

- Matlab2018a(程序设计软件)

3）操作系统(2 选 1)：

- Windows7
- Windows10

2、粒子群算法

蚁群算法是一种智能优化算法，通过蚁群优化求解复杂问题，ACO 在离散优化问题方面有比较好的优越性。蚁群算法是一种用来寻找优化路径的概率型算法。它由 Marco Dorigo 于 1992 年在他的博士论文中提出，其灵感来源于蚂蚁在寻找食物过程中发现路径的行为。

1）基本原理

单只蚂蚁的行为及其简单，行为数量在 10 种以内，但成千上万只蚂蚁组成的蚁群却能拥有巨大的智慧，这离不开它们信息传递的方式——信息素。

蚂蚁在行走过程中会释放一种称为“信息素”的物质，用来标识自己的行走路径。在寻找食物的过程中，根据信息素的浓度选择行走的方向，并最终到达食物所在的地方。信息素会随着时间的推移而逐渐挥发。

在一开始的时候，由于地面上没有信息素，因此蚂蚁们的行走路径是随机的。蚂蚁们在行走的过程中会不断释放信息素，标识自己的行走路径。随着时间的推移，有若干只蚂蚁找到了食物，此时便存在若干条从洞穴到食物的路径。由于蚂蚁的行为轨迹是随机分布的，因此在单位时间内，短路径上的蚂蚁数量比长路径上的蚂蚁数量要多，从而蚂蚁留下的信息素浓度也就越高。这为后面的蚂蚁们提供了强有力的方向指引，越来越多的蚂蚁聚集到最短的路径上去，蚂蚁路径示意图如图 1 所示：

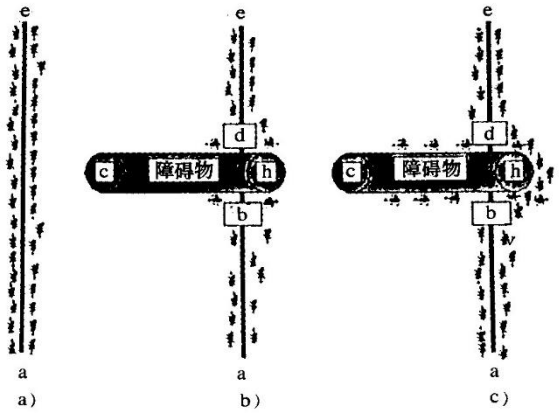

图 1.蚂蚁路径示意图

fig1. Ant path diagram

①高度结构化的组织——虽然蚂蚁的个体行为极其简单，但由个体组成的蚁群却构成高度结构化的社会组织，蚂蚁社会的成员有分

工，有相互的通信和信息传递。

②自然优化——蚁群在觅食过程中，在没有任何提示下总能找到从蚁巢到食物源之间的最短路径；当经过的路线上出现障碍物时，还能迅速找到新的最优路径。

③信息正反馈——蚂蚁在寻找食物时，在其经过的路径上释放信息素（外激素）。蚂蚁基本没有视觉，但能在小范围内察觉同类散发的信息素的轨迹，由此来决定何去何从，并倾向于朝着信息素强度高的方向移动。

④自催化行为——某条路径上走过的蚂蚁越多，留下的信息素也越多（随时间蒸发一部分），后来蚂蚁选择该路径的概率也越高，如图 2 所示：

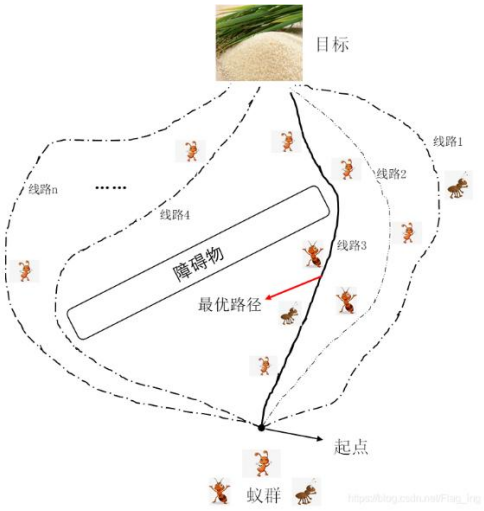

图 2.ACO 算法原理图

fig2.ACO Algorithm theory

2）算法思路

①根据具体问题设置多只蚂蚁，分头并行搜索。

②每只蚂蚁完成一次周游后，在行进的路上释放信息素，信息素量与解的质量成正比。

③蚂蚁路径的选择根据信息素强度大小（初始信息素量设为相等），同时考虑两点之间的距离，采用随机的局部搜索策略。这使得距离较短的边，其上的信息素量较大，后来的蚂蚁选择该边的概率也较大。

④每只蚂蚁只能走合法路线（经过每个城市 1 次且仅 1 次），为此设置禁忌表 p 来控制，如图 3 所示：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}(t)]^{\beta}}{\sum_{s\in J_k(i)}[\tau_{is}(t)]^{\alpha}\cdot[\eta_{is}(t)]^{\beta}}=\dfrac{X}{Y}, & \text{如果} j\in J_k(i)\\ 0, & \text{否则}\end{cases} \qquad \eta_{ij}=\frac{1}{d_{ij}}$$

图 3.状态转移矩阵更新公式

fig3. Update formula of state transition matrix

⑤所有蚂蚁都搜索完一次就是迭代一次，每迭代一次就对所有的边做一次信息素更新，原来的蚂蚁死掉，新的蚂蚁进行新一轮搜索。

⑥更新信息素包括原有信息素的蒸发和经过的路径上信息素的增加。更新公式如式（1）所示：

$$\tau_{ij}(t+n)=(1-\rho)\cdot\tau_{ij}(t)+\Delta\tau_{ij} \qquad (1)$$

⑦达到预定的迭代步数，或出现停滞现象（所有蚂蚁都选择同样的路径，解不再变化），则算法结束，以当前最优解作为问题的最优解。

3、例题与程序设计

1）例题

求解下列函数的最小值：

$$f(\mathrm{x}) = -x_1^4 + x_2^4 - \cos(3x_1) - 0.4 \times \cos(4 \times x_2) + 0.6 \qquad (2)$$

2）算法步骤：

算法流程图如图 4 所示：

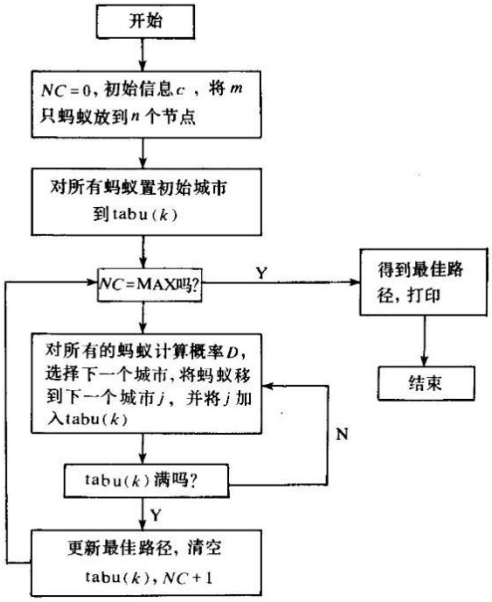

图 4.ACO 算法流程图

fig4. ACO algorithm flow chart

3）Matlab 代码：

在程序编写之前，对符号做出约束说明，如表 1 所示：

表 1.符号描述图表
table1.Symbol Description

Symbol	Description
Ant	蚂蚁的数量
Times	蚂蚁的移动次数
Rou	信息数的挥发系数
P0	转移概率常数

```
clear all
%初始化
Ant=300; % 蚂蚁的数量
Times=800; % 蚂蚁的移动次数
Rou=0.9; % 信息数的挥发系数
p0=0.2; % 转移概率常数
Lower_1=-1; % 设置搜索范围
Upper_1=1;
Lower_2=-1;
Upper_2=1;
for i=1:Ant
    X(i,1)=(Lower_1 + (Upper_1-Lower_1)*rand); % 设置随机位置
    X(i,2)=(Lower_2 + (Upper_2-Lower_2)*rand); % 设置随机位置
    Tau(i)=F(X(i,1),X(i,2)); % 计算目标函数值
end
step=0.05;
f='-(x.^4+3*y.^4-0.2*cos(3*pi*x)-0.4*cos(4*pi*y)+0.6)';

[x,y]=meshgrid(Lower_1:step:Upper_1, Lower_2:step:Upper_2);
z=eval(f); % z轴
figure(1);
subplot(1,2,1);
mesh(x,y,z); % 得到一个蜘蛛网
hold on;
plot3(X(:,1),X(:,2),Tau,'k*') % 绘制三维图
hold on;
text(0.1,0.8,1,'蚂蚁的初始位置');
xlabel('x');
ylabel('y');
zlabel('f(x,y)');
for T=1:Times
    lamda=1/T;
    [Tau_Best(T),BestIndex]=max(Tau); % 返回函数的最大值和其索引
    for i=1:Ant
        P(T,i)=(Tau(BestIndex)-Tau(i)) / Tau(BestIndex); % 计算状态转移概率
    end
    for i=1:Ant
        if P(T,i)<p0 % 局部搜索
            temp1=X(i,1)+(2*rand-1)*lamda;
            temp2=X(i,2)+(2*rand-1)*lamda;
        else
            temp1=X(i,1)+(Upper_1-Lower_1)-(rand*0.5);
            temp2=X(i,2)+(Upper_2-Lower_2)-(rand*0.5);
        end
        % 越界处理
        if temp1<Lower_1
            temp1=Lower_1;
        end
        if temp1>Upper_1
            temp1=Upper_1;
        end
        if temp2<Lower_2
            temp2=Lower_2;
        end
        if temp2>Upper_2
            temp2=Upper_2;
        end
        if F(temp1,temp2)>F(X(i,1),X(i,2)) % 判断蚂蚁是否移动
            X(i,1)=temp1;
            X(i,2)=temp1;
        end
    end
    for i=1:Ant
        Tau(i)=(1-Rou)*Tau(i)+F(X(i,1),X(i,2));
    end
end
subplot(1,2,2);
mesh(x,y,z);
hold on;
x = X(:,1);
y = X(:,2);
plot3(x,y,eval(f),'k*'); % 绘制三维图
hold on;
text(0.1,0.8,1,'蚂蚁的最终分布位置');
xlabel('x');
ylabel('y');
zlabel('f(x,y)');
```

图 5.ACO 算法实现

fig5. ACO algorithm implementation

4）函数调用方式：

运行代码 aco.m，运行结果如图 6 所示：

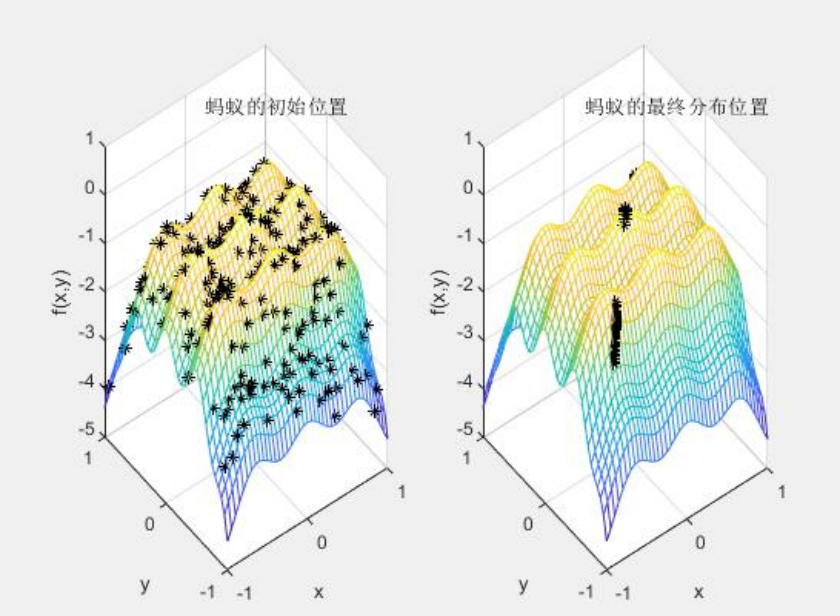

图 6.ACO 算法结果

fig5. ACO algorithm result

	1	2	3	4	5	6	7	8	9	10
1	-124.8620	-138.6280	-135.1052	-33.3182	-199.0521	-155.4683	-61.0859	-99.4686	-128.7471	-145.0960
2	-124.8620	-138.6280	-43.3706	-33.3182	-31.0990	-155.4683	-61.0859	-99.4686	-128.7471	-145.0960
3	-155.4460	-134.8708	-79.6924	-75.6966	-36.9214	-348.6853	-285.5257	-223.5335	-288.9671	-267.1431
4	-640.9570	-503.2516	-590.4037	-1.2020e+...	-258.2674	-1.0452e+...	-2.2574e+...	-770.1258	-584.4429	-1.9665e+...
5	-5.3726e+...	-4.1311e+...	-5.3830e+...	-1.1549e+...	-1.4176e+...	-4.6938e+...	-1.6867e+...	-1.3881e+...	-3.8926e+...	-2.7635e+...
6	-2.8387e+...	-2.1777e+...	-2.8694e+...	-3.2234e+...	-7.0568e+...	-2.2441e+...	-2.9472e+...	-7.7650e+...	-1.9983e+...	-1.5170e+...
7	-5.4386e+...	-3.8370e+...	-5.0615e+...	-5.1685e+...	-1.2360e+...	-3.9129e+...	-4.1325e+...	-1.3761e+...	-3.9975e+...	-2.6835e+...
8	-4.6016e+...	-4.1539e+...	-5.4801e+...	-1.2008e+...	-1.3372e+...	-4.2316e+...	-4.3589e+...	-1.4907e+...	-944.4791	-2.5372e+...
9	-4.5103e+...	-5.6067e+...	-5.5258e+...	-1.3311e+...	-1.3483e+...	-4.2664e+...	-4.3836e+...	-1.5032e+...	-611.1660	-2.5212e+...
10	-4.5010e+...	-8.4387e+...	-4.2676e+...	-1.3443e+...	-6.4635e+...	-4.2699e+...	-4.3861e+...	-1.5044e+...	-577.5286	-2.3597e+...
11	-4.5001e+...	-5.0605e+...	-4.1417e+...	-1.3456e+...	-2.1169e+...	-4.2673e+...	-4.3863e+...	-1.5046e+...	-574.1618	-1.7317e+...
12	-4.5000e+...	-4.7226e+...	-5.1609e+...	-1.1063e+...	-9.7517e+...	-5.5899e+...	-7.6193e+...	-1.1104e+...	-573.8251	-1.8031e+...
13	-4.5087e+...	-4.6888e+...	-5.2629e+...	-1.0824e+...	-8.6100e+...	-5.7222e+...	-7.9426e+...	-1.6462e+...	-573.7914	-1.8103e+...
14	-4.1845e+...	-4.6855e+...	-5.2731e+...	-1.0800e+...	-8.4958e+...	-5.7354e+...	-8.3755e+...	-1.6998e+...	-573.7881	-1.8110e+...
15	-4.4613e+...	-5.1880e+...	-7.1044e+...	-1.0797e+...	-8.4844e+...	-5.7367e+...	-8.4188e+...	-1.7051e+...	-573.7877	-1.6500e+...
16	-6.3269e+...	-5.9026e+...	-5.6545e+...	-1.0797e+...	-8.4833e+...	-5.7368e+...	-8.4231e+...	-2.1328e+...	-573.7877	-1.6262e+...
17	-9.0915e+...	-5.9740e+...	-5.5096e+...	-1.3417e+...	-7.7412e+...	-5.7369e+...	-1.2304e+...	-2.1755e+...	-573.7877	-1.6238e+...
18	-9.3679e+...	-5.9812e+...	-6.6404e+...	-1.3679e+...	-7.6670e+...	-4.3283e+...	-1.0866e+...	-2.1798e+...	-573.7877	-1.6236e+...
19	-9.3956e+...	-5.6201e+...	-6.5321e+...	-1.5590e+...	-7.6595e+...	-4.1461e+...	-1.0722e+...	-2.1802e+...	-573.7877	-1.6236e+...
20	-1.1933e+...	-5.5840e+...	-8.5874e+...	-1.3557e+...	-7.6588e+...	-4.2887e+...	-1.2527e+...	-1.9674e+...	-573.7877	-1.6236e+...
21	-1.3264e+...	-5.7851e+...	-9.1154e+...	-1.3257e+...	-9.8067e+...	-4.4608e+...	-1.3269e+...	-1.8159e+...	-594.8733	-1.5434e+...

图 7.ACO 算法部分状态转移矩阵计算结果

fig7. ACO algorithm partial state transition matrix calculation results

经过程序计算可知，目标函数的最大值为 0.8215284206082081。aco.m 程序为主函数，F.m 程序为目标函数值。