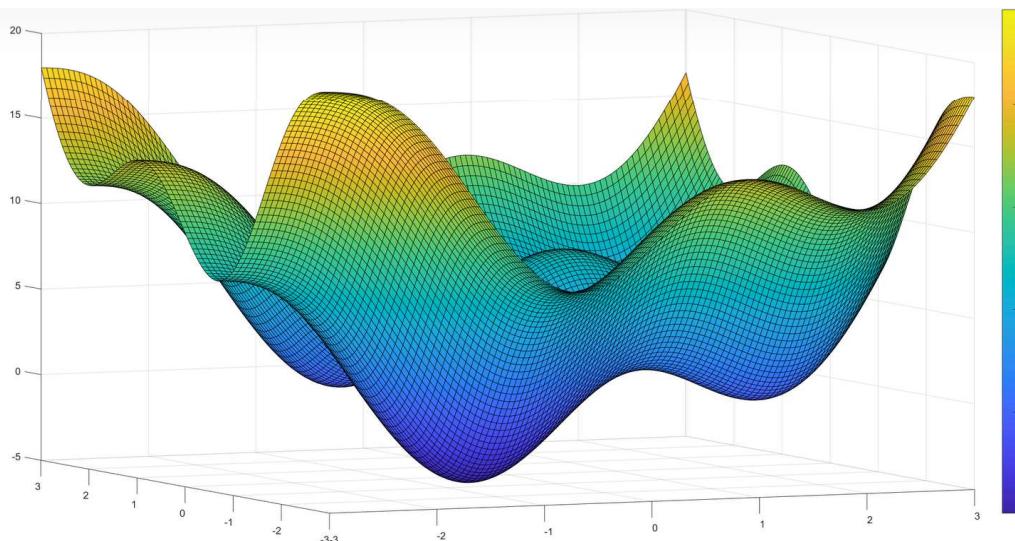


知乎

首发于
大学牲们的初学者教程

...

写文章



赞同 3

分享

遗传算法及用遗传算法解决旅行商问题（超详细）——基于 Matlab



DDKM

关注他

3 人赞同了该文章

本文适合没有学习过遗传算法的小白，也欢迎大佬进来指正

本文侧重通过旅行商这个问题详细的介绍遗传算法。作者本人也刚刚学习了遗传算法，通过费曼学习法巩固一下。

本文的第一部分参考了B站up主“freexyn”的教程，讲得很详细，大家可以看一看，这个up主还有很多其他精彩的课程。

freexyn|Matlab47: 遗传算法_哔哩哔哩
 _bilibili
www.bilibili.com/cheese/play/ep83978?cs...



本文第二部分的例题来自B站up主“数模加油站”的视频。

[https://www.bilibili.com/video/BV1EK41187QF?p=38&vd_sou...](https://www.bilibili.com/video/BV1EK41187QF?p=38&vd_source=d9b9d6ab2a320e2c549cedb1...)

对于第一部分遗传算法介绍，主要介绍思路，因为直接求函数的最优点在matlab中有现成的工具。

对于第二部分用遗传算法解旅行商问题，会详细介绍我的做法的每一步，会附上matlab的代码。对遗传算法已经有一定了解的同学可以直接看这部分。最后也会附上百度网盘下载整个程序压缩包的链接。

一、遗传算法介绍

我们先来了解一下遗传算法的原理。

▲ 赞同 3 ▾ 添加评论 分享 喜欢 收藏 申请转载 ...

知乎

首发于
大学牲们的初学者教程

从问题出发，假如现在有一个函数 $f(x, y) = x^2 + y^2 + 3\sin(2x) + 3\sin(2y)$ ，这个函数很简单，是为了举例而用，实际的问题可能会非常复杂。

我们要求它的最小值是多少，取到最小值时， x 和 y 的值分别是多少。或许你会用偏导来求得精确答案，这个函数的偏导确实很容易求。

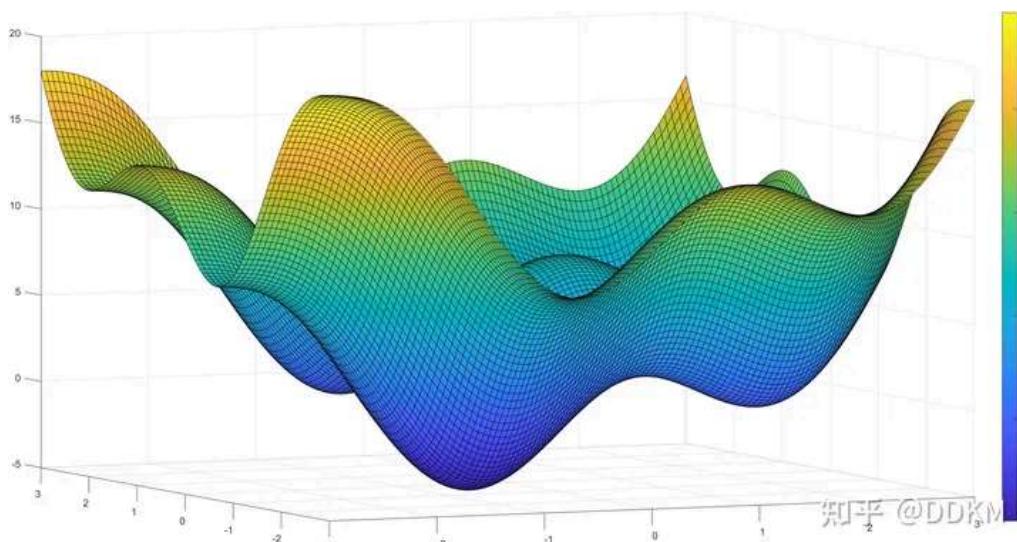
关于 x 的偏导： $2x + 6\cos(2x)$

关于 y 的偏导： $2y + 6\cos(2y)$

但是偏导为0的方程组仍然不好解，还是要用数值法解，而且不太可能得到精确解。何况当函数变得更复杂时，连求偏导本身都会变得不可行。

所以我们考虑直接用数值法来求解。

我们最容易想到的就是近似穷举的做法：大概限定一个 x 和 y 的范围。比如这题，取到最小值时， x 和 y 的绝对值显然不可能超过3。那就取 $x, y \in [-3, 3]$ 的这个平面内，按步长为0.05来取点，总共取 121×121 个点，然后找出最小值，这样就找到了近似解。这是个自然的想法，我们用计算机很容易实现，下面是在matlab中的模拟：



$f(x, y) = x^2 + y^2 + 3\sin(2x) + 3\sin(2y)$ 在 $[-3, 3]$ 内的图像

这时，当 $x = -0.65, y = -0.65$ ，取到近似最小值， $f(x, y) = -4.9363$ 。

看起来不错，但是这个方法有计算的局限性：

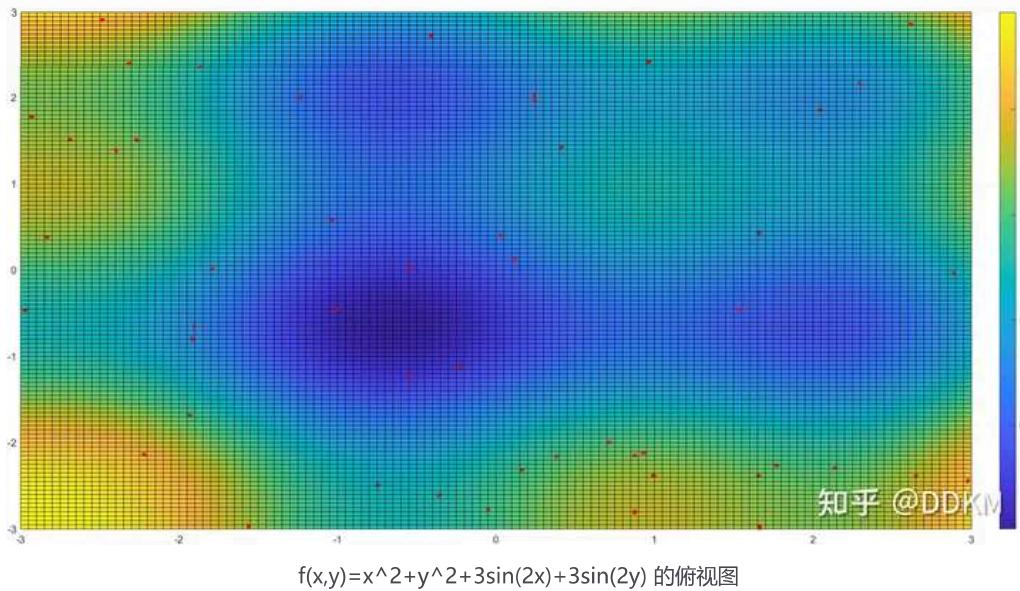
- 这个近似解的精度不怎么样，如果想提高精度，计算量会成倍增长，例如将0.05改为0.025，计算量翻了四倍。在这个例子里，计算机当然都能秒出答案，但是我们真正需要应用的是非常复杂的实例，计算量的倍数增长会使求解成本大大提高。之后在旅行商问题中会看到这一点。
- 这个计算方法花费了大量算力在完全没有用的地方。我们可以看到，在 x 和 y 绝对值大于1的地方的计算都是无用功，因为最小值不在这个范围附近。这还是在我们提前框定了 $[-3, 3]$ 的前提下，更常见的情况是人根本无法提前找到一个合适的包含最优解的范围。

到这里我们可以引入更有效的算法了，就是启发式算法中的遗传算法。

2. 什么是遗传算法

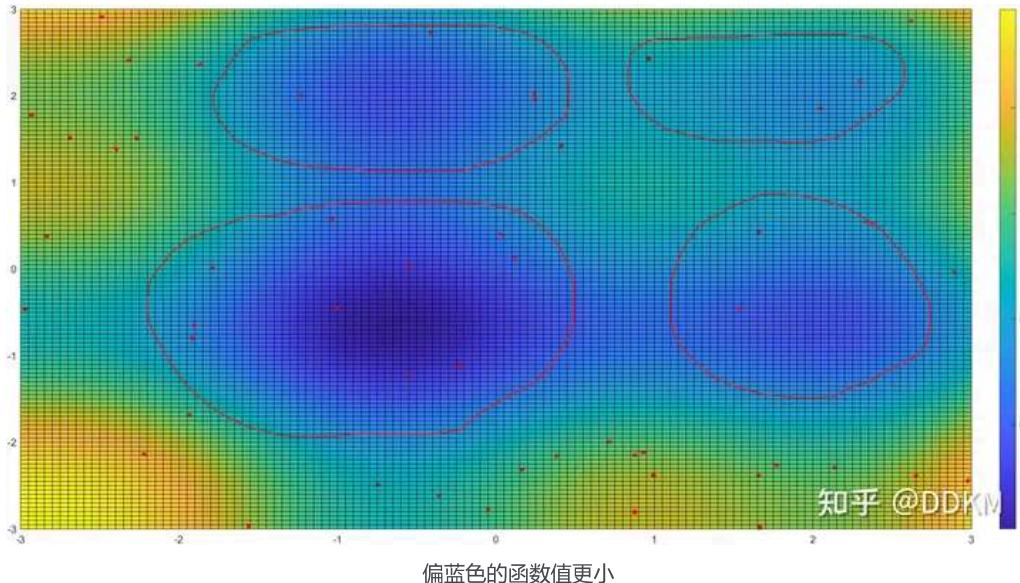
启发式算法有很多，例如模拟退火算法，蚁群算法，遗传算法。这些计算方法大多用于优化，也就是用比穷举少得多的算力求一个比较好的可以被接受的较优解，而不是最优解。这些算法的灵感来自于一些生物物理现象，因此被称为启发式算法，从他们的名字中就可以看到灵感的来源。

知乎

首发于
大学牲们的初学者教程

上面这张俯视图上的红点就是随机选取的点。我们算出这五十个点的值。接下来选取一些比较好的值。

不妨认为我随便圈的这些值是较好的值。

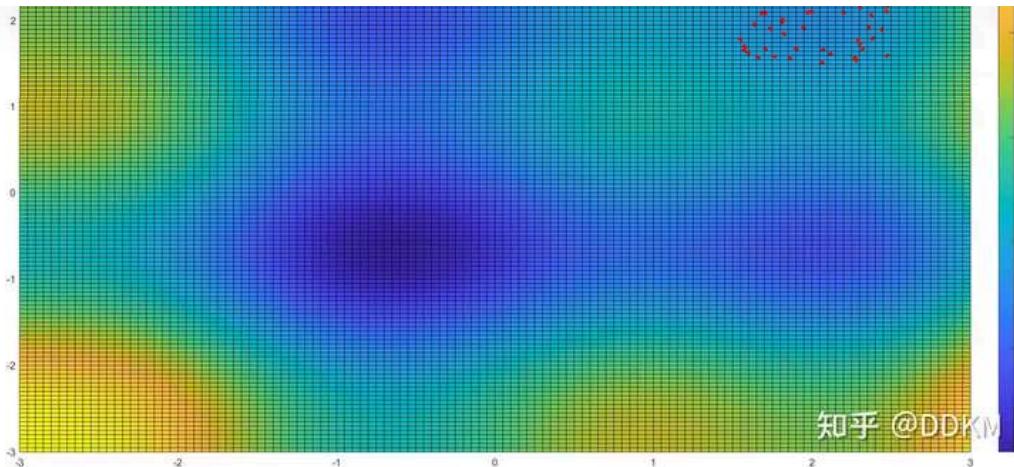


下一步我们要用这些值去生成一些新的值，这里就简单一些，取平均数吧。在较好的值中，随机挑选两个，找到它们的中点，用这样的方法重新生成五十个点。然后不断重复这个选择，生成新的值的过程。你一定可以想象到，最后这些点都将逐渐收敛到图中的四个局部最小值附近。我们最终将可以得到左下角这个局部最小值（同时也是全局最小值）最为我们的结果。

这时候聪明的同学就要问了，如果一开始的初始值是下面这个图该怎么办？



首发于
大学牲们的初学者教程



这张图当然是我手动调整的，随机数很难出现这种情况。但是在复杂的问题中，确实很可能初始值得到我们不希望看到的样子，或者在中间某一步变成这样。什么样子呢？所有的点都分布在局部最小值附近了。那么按照我们刚才的取中点的做法，最终结果无论如何都只能得到右上角这个局部最小值，而不可能得到左下角的全局最小值了，我们的遗传算法也就失败了。

为了解决这个问题，我们还需要引入一些不确定因素。每一次迭代中，我们都挑选一些点，通过这些点随机地生成另外一些点，那么这部分生成的点就能够突破这个局部最小值的陷阱，很有可能来到全局最小值的附近，这样就能最终收敛到全局最小值。

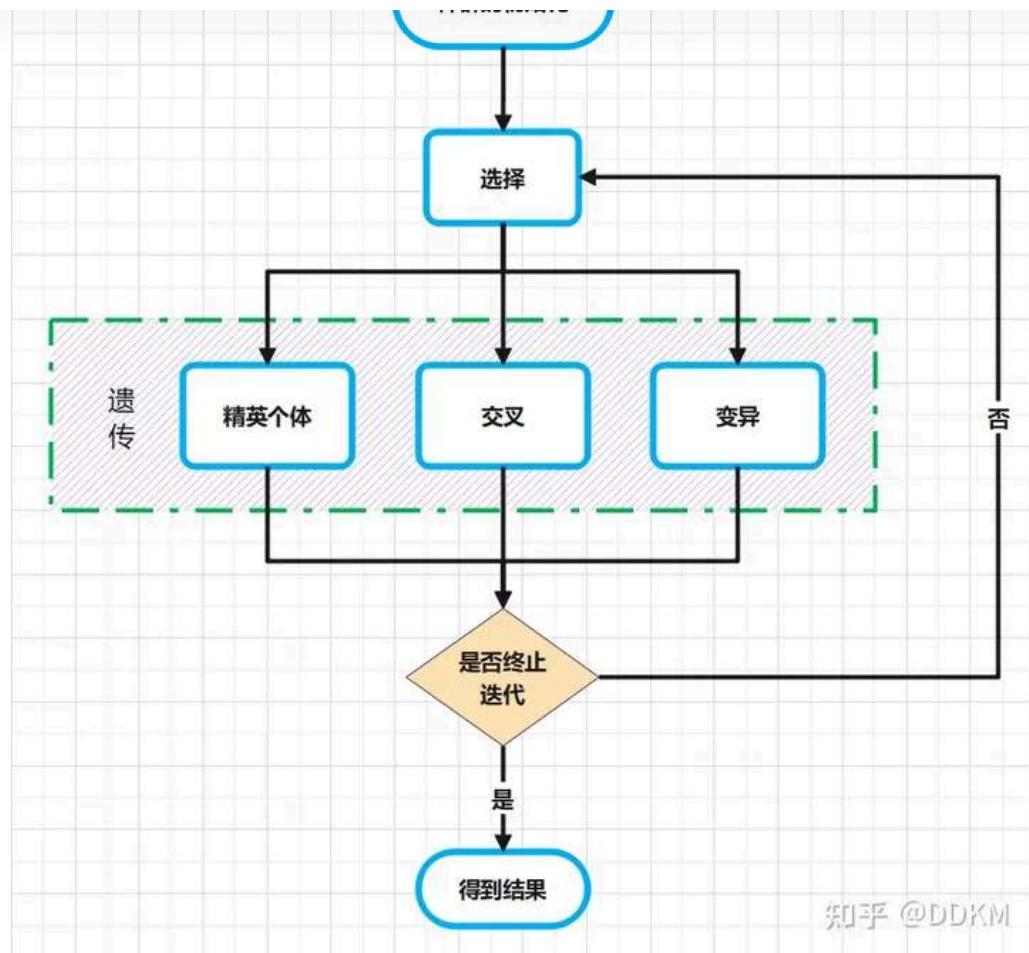
以上是一个遗传算法的超级简化版本。

接下来我们正式开始。

3. 遗传算法的一般过程

遗传算法实在模拟自然界生物种群的遗传，自然选择和进化的过程。它的主要步骤包括初始化种群，选择，遗传（包括了精英，交叉，变异）和终止。在我们上面的简化版例子中，选择较优的点就相当于选择，取两个点的中点就相当于交叉，通过一些点随机地生成另外一些点就相当于变异，注意这里只是相当于，实际的过程并不是这样的。另外，精英和终止没有提到。下面是一张流程图。

知乎

首发于
大学牲们的初学者教程

知乎 @DDKM

遗传算法流程图

我们将分步作详细介绍，这里先介绍一些概念：

• 种群

是指每一次迭代的这些点的集合或者说矩阵。在上面的简单例子中，种群就是 50×2 的矩阵。 x 和 y 各五十个。这个种群在迭代过程中不断地更新变化，但是它的数量是保持不变的。

• 个体

是指种群中的某一个点。

• 精英个体

是指函数值最小的那一部分点，具体是最小的那个，还是最小的那十个（精英率），根据不同的情景由你调整。

• 交叉个体

由某两个父代通过一定的方法形成的子代，这样形成的子代往往保留的父代的优良性状，也就是说，这个子代的函数值也有很大的可能是很小的。在上面的例子中，我们用了取中点的方法，这是简化了的方法，是不可靠的，之后会讲到一般的方法。

• 变异个体

由某一个父代通过一定的方法形成的子代，主要是为了避免落入局部最小值的陷阱，保证了种群的多样性。

• 精英率、交叉率、变异率

$$\text{精英率} = \frac{\text{精英个体数量}}{\text{种群数量}}, \text{ 交叉率} = \frac{\text{下一代中由交叉产生的个体数量}}{\text{种群数量}}, \text{ 变异率} = \frac{\text{下一代变异的个体数量}}{\text{种群数量}}$$

来举一个例子。在上面的例子中，种群数量为50，我们不妨设置精英率为2%，交叉率设置为



首发于
大学牲们的初学者教程

这三个值具体的取法会再后面遗传的部分详细介绍。

• 收敛

种群不断靠近局部最小值的过程就是收敛的过程。当种群很接近局部最小值时，每个个体之间的差异会很小（这很容易想象），这时种群就收敛了。

• 失速

在遗传算法的过程中，种群当中的最优值没有发生变化，称为失速。



（1）种群的初始化

这一步要考虑以下一些问题。

• 种群数量是多少

一般来说，种群数量越大，最后得到的结果也会比较好。但是随着种群数量的增大，计算成本也会显著增加，因此这个值并不是越大越好。综合考虑你的计算资源（包括算力，能够接受的计算时间等），结果所需要的精度，取到适合的值就可以。

• 种群的初始值应该怎么选

最简单的方法就是像我们上面所做的一样，完全随机。但是一个好的初始状态能够让我们的大大节约计算资源，并且结果也会好得多。比如说，我们对上面这题有一个大概的估计，最优解的x和y都应该落在[-1,0]之间，那么我们就可以把初始种群定为[-1,0]之间的随机数-rand(50,2)，从而使结果更好，计算时间更短。

在我们后面的旅行商问题中，我们会用贪心算法来初始化，这个在后面会具体解释。

（2）选择

这一步要完成的任务就是在上一代的种群中，根据精英率，交叉率和变异率，选出一些父代来分别作为精英个体，交叉和变异的父代。

在选择的时候我们希望我们的子代通过遗传的精英，交叉和变异之后，既要能够继承父代种群的优良性状，能够继续找到更小的函数值，由希望种群有一定的多样性，不要所有个体落入同一块区域，这样很有可能导致结果掉入局部最小值的陷阱。因此我们会用不同的方法来分别为选择用于精英个体的父代，用于交叉的父代和用于变异的父代。

• 精英个体

这个没什么好说的，函数值最小的那几个就是了，具体的个数为 种群数量 × 精英率。

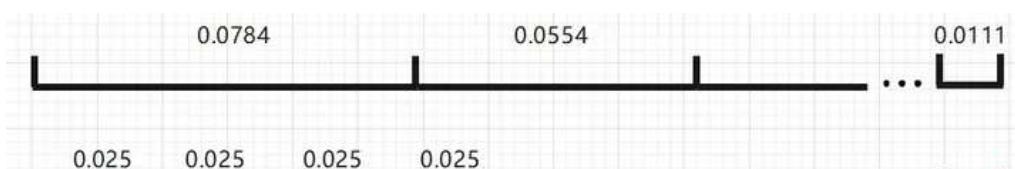
• 交叉个体

这个的选择方法很多，这里只介绍一种常见的方法，有适应度缩放的随机均匀法。这对于绝大多数的情况都够用了。

还是以种群数量为50举例。

首先将种群进行排序，每个个体分配到一个序号，1到50。最优的点是1，最差的点是50。将它们开根号后取倒数，组成一个新的序列 $1, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}, \dots, \frac{1}{\sqrt{50}}$ ，将这一个数列归一化（每个值除以它们的总和），就可以得到每个个体被取到的概率为 0.0784, 0.0554, ..., 0.0111。

将归一化后的数值作为线段分布在0到1的数轴上，然后均匀作40个点（按照上面交叉个体数为40的例子），这个点落在哪一个范围内，就取哪一个父代。





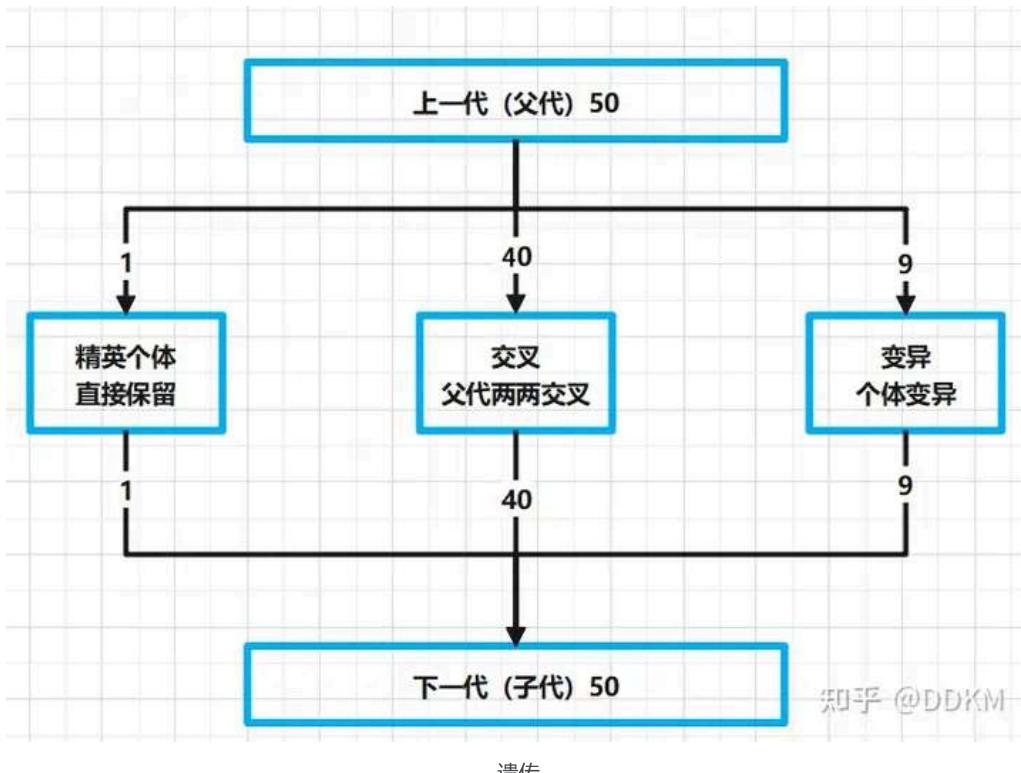
首发于
大学牲们的初学者教程

个体后的交配机会产生后代的概率更大。这能够保证我们的种群朝着更优的方向进化。

- 变异个体

因为变异个体的目的是保持种群的多样性，所以变异完全可以随机在50个种群中选择9个。

(3) 遗传



遗传

知乎 @DDKM

- 精英

这个直接保留上一代的就可以。精英率应该怎么来确定。精英是为了至少不丢失目前的最优解，这样以来遗传过程无论如何不会倒退。但是如果精英率太高，就很容易过早的收敛，或者失速。

- 交叉

交叉的方法可以说是遗传算法最重要的部分。有太多的交叉方法可供选择，对于不同的情景，我们也要运用不同的交叉方法。对于交叉方法的选择，我们需要清楚交叉的目的是保留种群的优良性状。

三 目录 与三步：编码，交叉，解码。

这里举一个很简单的例子来说明。我们用转化为二进制后，片段交叉的方法。我想交叉父代 $(-0.6, -0.4)$ 和 $(-0.5, -0.5)$ 。首先交叉x，将-0.6和-0.5转换为二进制后，挑选一个位置交换量者从这个位置开始的片段，构成新的二进制数码后再解码为十进制数。对y也这样操作。这样得到了两个新的子代。

编码和交叉的方式非常多，针对不同的问题也有不同的适用方法。这里不再介绍更多，可以在网上搜教程。

- 变异

变异的目的是保持种群的多样性，避免种群过早收敛，避免种群陷入局部最优解的陷阱。举一个简单的例子，将一个数转换为二进制以后，随便挑选一位将0变成1或者1变成0，再转换为十进制即可。



首发于
大学牲们的初学者教程

我们要判断什么时候终止迭代，通常来说有这么几种判据：

- **计算资源类**

主要包括迭代次数和运算时长。可以提前设置，在达到一定的条件以后即停止。

- **达到优化目标类**

主要包括目前求解的最优值已经小于了预期所需要的值和已经收敛。

- **计算失败类**

主要包括连续失败达到一定轮次。

这些判断的标准也是自由决定的，可以根据实际问题而改变。

二、用遗传算法解决旅行商问题

接下来进入正题。首先看题目：有一位商人要拜访100个城市，他的起点已经确定（起点不算在这100个城市内）。这位旅行商必须拜访所有的城市（所有城市的经纬度坐标都已经给出，在txt文件中），并且不能到同一个城市两次，在最后，他必须回到起点。我们的目标是规划出一条路线，要求总路程最短。

不妨叫它们1号城市到100号城市，起点记为0号。在这道题中，每一个路线就是一个100个数的排列。总共有 $100!$ 种路线（ $100!$ 大约是 9.3×10^{157} ），穷举是一般计算机不可能完成的任务。

个人感觉超级计算机应该也不太行。

所以我们尝试用遗传算法来解决这个问题。

(1) 确定一些参数

- 种群数量：100
- 精英率：1%
- 变异率：1%
- 交叉率：80%

还有其他一些准备工作一并放在这里（主脚本1）。

```
%> Initialize
clc, clear, close all
sj0=load('TSP.txt');
x=sj0(:,1:2:8); x=x(:);
y=sj0(:,2:2:8); y=y(:);
sj=[x y]; d1=[70,40];
xy=[d1;sj;d1]; sj=xy*pi/180;
d=zeros(102);
for i=1:101
    for j=i+1:102
        d(i,j)=6370*acos(cos(sj(i,1)-sj(j,1))*cos(sj(i,2))*...
            cos(sj(j,2))+sin(sj(i,2))*sin(sj(j,2))); %球面上的距离公式
    end
end
d=d+d'; % Distance table

n_city=101; % 城市数量
path=zeros(n_city-1,n_city); long=zeros(n_city-1,1); %巡航路径及长度初始化
```



首发于
大学牲们的初学者教程

```
function [path_sorted, long_sorted] = Sorting(path,long)
    [long_sorted, sortedIndices] = sort(long);
    path_sorted = path(sortedIndices, :);
end
```



辅助的画图函数

```
function [] = Draw(xy,path_best)
    x=xy(1:101,1);
    y=xy(1:101,2);
    figure;
    scatter(x,y,25,'filled')
    for i = 1:length(x)
        text(x(i), y(i), num2str(i),"FontSize",10);
    end
    hold on;
    plot(x(path_best),y(path_best),'r')
end
```

辅助的距离函数

```
function [long] = distance(path,d)
    long=0;
    l=length(path);
    for i=1:l-1
        long=long+d(path(i),path(i+1));
    end
    long=long+(d(path(101),102));
end
```

辅助的检查函数（可不用）

```
function [test_out] = Check(matrix)
    [x,y]=size(matrix);
    exam=zeros(x,y);
    test_out=zeros(x,1);
    for i=1:x
        for j=1:y
            exam(i,matrix(i,j))=exam(i,matrix(i,j))+1;
        end
        test_out(i)=prod(exam(i,:));
    end
end
```

(2) 初始化

好的初始化能够大大提升效率并提高准确度。在旅行商问题中我们可以使用贪心算法作为我们的初始化方法。贪心算法一句话就可以解释：每一步都取接下来的最优解。在我们的实例中，我们分别



首发于
大学牲们的初学者教程

```

path(i,1)=1;
path(i,2)=i+1;
end

for j=2:n_city % 循环索引，同时也是初始位置
    dt=d;
    dt(:,1)=Inf; dt(:,j)=Inf; dt(dt == 0) = Inf;
    current=j; % current 当前的位置
    for i=3:101
        [minValue, colIndex] = min(dt(current,1:101));
        path(j-1,i)=colIndex;
        long(j-1)=long(j-1)+minValue;
        dt(:,colIndex)=Inf;
        current=colIndex;
    end
    long(j-1)=long(j-1)+d(1,j)+d(current,1);
end
[value,index]=min(long);
% Draw(xy, path(index,:));

```

(3) 选择

对于交叉的选择，采用之前提到了有适应度缩放的随机均匀法。

对于变异的选择，直接在100个中完全随机挑选。

```

% 准备工作，计算选择分布
for i=1:100
    scale(i)=1/(i^(1/2));
end
scale=scale./sum(scale);
scale=cumsum(scale);
point=(0.0125:0.0125:1);
Select_PMX=zeros(40,1);
Select_OX=zeros(40,1);
p=1;
for i=1:80
    for j=p:100
        if point(i)<=scale(j) && rem(i,2)==1
            Select_PMX(i/2-0.5+1)=j;
            p=j;
            break;
        elseif point(i)<=scale(j) && rem(i,2)==0
            Select_OX(i/2)=j;
            p=j;
            break;
        end
    end
end
path0=path;
long0=long;

```



首发于
大学牲们的初学者教程

```
function [Elitism, Mutation, Crossover_PMX, Crossover_OX] = Select(path, Select_PMX, S
Elitism=path(1,:);

temp=randperm(100,19)';
Mutation=path(temp,:);

Crossover_PMX=path(Select_PMX,:);
randomIndices = randperm(40)';
Crossover_PMX=Crossover_PMX(randomIndices,:);

Crossover_OX=path(Select_OX,:);
randomIndices = randperm(40)';
Crossover_OX=Crossover_OX(randomIndices,:);
end
```

(4) 交叉

旅行商问题无需编码。100个数的一个排列本身就是一串编码。

在本题中，我采用了两种交叉方法，部分匹配交叉（PMX）和顺序交叉（OX）。对于这两种交叉方法的详细介绍，推荐大家看这个。

我无论怎么用文字表达，都不会有这篇文章讲得更清楚。这篇文章包含大部分的交叉方法。这些变异方法用在旅行商问题中容易带来一种错觉：就是子代似乎没有能继承父代的优良性状。但其实是继承了的，而且它的继承程度不小，导致收敛速度比想象的更快。我再不断调小精英率，提高变异率之后才将其收敛速度调到合适的程度。

PMX函数的代码

```
function [child] = PMX(parent1, parent2)
% 获取路径长度
n = 101;

% 随机选择交叉点
random=sort(randperm(100,2));
crossPoint1 = random(1)+1;
crossPoint2 = random(2)+1;

% 初始化子代为零
child1 = zeros(1, n);
child2 = zeros(1, n);

% 从一个父代复制交叉段到相应的子代
child1(crossPoint1:crossPoint2) = parent1(crossPoint1:crossPoint2);
child2(crossPoint1:crossPoint2) = parent2(crossPoint1:crossPoint2);
```



首发于
大学牲们的初学者教程

```

child1(i) = parent2(i);
else
    child1(i) = mapCity(parent2(i), parent1, parent2, child1);
end

if ~ismember(parent1(i), child2)
    child2(i) = parent1(i);
else
    child2(i) = mapCity(parent1(i), parent2, parent1, child2);
end
end
child=[child2;child1];
end

function city = mapCity(mappedCity, parent1, parent2, child)
% 辅助函数：处理城市映射
% 映射冲突的城市到新的城市
while ismember(mappedCity, child)
    mappedIndex = parent1 == mappedCity;
    mappedCity = parent2(mappedIndex);
end
city = mappedCity;
end

```

OX函数的代码

```

function [child] = OX(parent1, parent2)
% 获取路径长度
n = 101;

% 随机选择交叉点
random=sort(randperm(100,2));
crossPoint1 = random(1)+1;
crossPoint2 = random(2)+1;

% 初始化子代
child1=zeros(1,n);
child2=zeros(1,n);

% 从一个父代复制交叉段到相应的子代
child1(crossPoint1:crossPoint2) = parent1(crossPoint1:crossPoint2);
child2(crossPoint1:crossPoint2) = parent2(crossPoint1:crossPoint2);

% 顺序填充剩余的城市
iter=[1:crossPoint1-1,crossPoint2+1:n];
iter_i=1;
for i=1:n
    if ~ismember(parent1(i),child2)
        child2(iter(iter_i))=parent1(i);
        iter_i=iter_i+1;
    end
end

```



首发于
大学牲们的初学者教程

```

        iter_i=iter_i+1;
    end
end

child=[child1;child2];
end

```



(5) 变异

这题中，我也采用了两种变异方法，分别是交换变异（Swap）和随机打乱变异（Scramble）。
交换变异很好理解，就是随机交换两个城市在链路上的位置。随机打乱变异则是选取某一段，在这一段中随机打乱里面的城市的位置。

Swap的函数代码

```

function [child] = Mut_Swap(parent)
    random=sort(randperm(100,2));
    SwapPoint1 = random(1)+1;
    SwapPoint2 = random(2)+1;
    temp=parent(SwapPoint1);
    parent(SwapPoint1)=parent(SwapPoint2);
    parent(SwapPoint2)=temp;
    child=parent;
end

```

Scramble函数的代码

```

function [child] = Mut_Scramble(parent)
    n=length(parent);
    random=sort(randperm(100,2));
    ScramblePoint1 = random(1)+1;
    ScramblePoint2 = random(2)+1;
    l=ScramblePoint2-ScramblePoint1+1;
    RandIndex=randperm(l);
    Scramble=parent(ScramblePoint1:ScramblePoint2);
    Scrambled=Scramble(:,RandIndex);
    child=[parent(1:ScramblePoint1-1),Scrambled,parent(ScramblePoint2+1:n)];
end

```

(6) 终止条件

先放上主脚本2

```

for m=1:3
    [path, long]=Sorting(path, long);
    path_best=path(1,:);
    long_best=long(1);
    iter_number=0;
    stall_number=0;

```



首发于
大学牲们的初学者教程

```
[Elitism, Mutation, Crossover_PMX, Crossover_OX]=Select(path, Select_PMX, Select_OX);

% PMX
for i=1:2:39
    path(i+20:i+21,:)=PMX(Crossover_PMX(i,:),Crossover_PMX(i+1,:));
end

% OX
for i=1:2:39
    path(i+60:i+61,:)=OX(Crossover_OX(i,:),Crossover_OX(i+1,:));
end

% Mutation_Inverse
for i=1:10
    path(i+1,:)=Mut_Inverse(Mutation(i,:));
end

% Mutation_Scramble
for i=1:9
    path(i+11,:)=Mut_Scramble(Mutation(i+10,:));
end

% CheckPoint
% test_out=Check(path);

% Change Long
for i=1:100
    long(i)=distance(path(i,:),d);
end

% Compare
[path, long]=Sorting(path, long);
if long_best>long(1)
    difference=long_best-long(1);
    path_best=path(1,:);
    long_best=long(1);
    stall_number=0;
else
    stall_number=stall_number+1;
    difference=-1;
end

if difference>0 && difference<10
    difference_number=difference_number+1;
else
    difference_number=0;
end
iter_number=iter_number+1;

%Stop
if iter_number>=20000 % 迭代次数
    stoplabel=1;
    break;

```



首发于
大学牲们的初学者教程

```

stoplabel=2;
break;
elseif difference_number>=10 % 连续最优解差值小于10次数
stoplabel=3;
break;
end
end
disp('遗传次数: ')
disp(m);
disp('stoplabel=')
disp(stoplabel);
disp('iter_number=')
disp(iter_number);
disp(long_best);

if m==3
break;
end
path=path0;
long=long0;
[path, long]=Sorting(path, long);
path(100,:)=path_best;
long(100,:)=long_best;

end
Draw(xy,path_best);

```



在这个实例中，我设置了三个终止条件：

- 迭代次数超过20000次，终止迭代，终止代码1。
- 连续失速超过5000次，终止迭代，终止代码2。
- 连续10个迭代最优解差值大于0但小于10，终止迭代，终止代码3。

由于旅行商问题的离散特性，实际上大部分迭代都是由2终止的。

最后可以画图看一看结果（图中没有画出最后一个城市回到起点，但实际是考虑了的）。



这个解的结果是 4.0814×10^4 KM，还算不错。我计算出过的最好的解大概是 4.03×10^4 KM。

三、更多技巧

1. 遗传算法有大量参数需要调试，包括种群数量，三个率，编码方法，交叉方法，变异方法，终止条件以及它们的组合。有时候刚开始的直觉并不准确，需要有耐心地调参。
2. 可以将几次遗传算法得到的结果作为初始值，再次带入遗传算法计算，相当于输入了一个很优的初始值。这或许可以进一步提高精度。但是这个方法在旅行商问题中基本没用。
3. 并不是每一步都要遵守现成的方法。有时候可以结合具体问题创新。例如在这题中，我们或许可以提出更好的交叉方法：我们模仿人在解决这类问题时的思路。人可以先把这一百个点分成几个“岛”，每个岛内先连好，再把岛连起来。我们考虑当某一段路程特别大时，那就有可能是在出岛或入岛，我们可以保持岛内的顺序大致不变，改变出岛入岛的位置。或者不改变岛的相对位置，修改岛内的顺序。这样是不是有可能更好地保留父代的优良形状。
当然上述方法只是一个不成熟的想法，没有经过实践，有兴趣的同学可以尝试。

完整程序百度网盘链接，提取码1145。

感谢阅读。

编辑于 2024-02-09 03:20 · IP 属地浙江



首发于
大学牲们的初学者教程

遗传算法 数学建模竞赛 优化



还没有人赞赏，快来当第一个赞赏的人吧！



欢迎参与讨论



发布



还没有评论，发表第一个评论吧

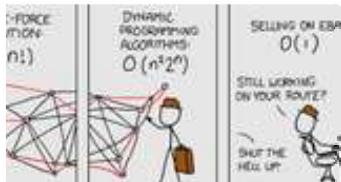
文章被以下专栏收录



大学牲们的初学者教程

一些适合大学牲们的详细初学者教程。

推荐阅读



基于遗传算法实现旅行商问题

zhouB...

发表于人工智能



“旅行商问题”太棘手？用图神经网络寻找最优解

读芯术



遗传算法解决旅行商 (TSP) 问题附Matlab代码

羊羊羊四只羊

遗传算法_旅行商(TSP)

- 什么是遗传算法模拟自然界中的遗传，种群优胜劣汰，一代更比一代强。种群中的个体差异就是基因差异。遗传算法就是模拟基因这个概念，将待解决问题的答案进行编码；答案最开始五花八门...

小木木