用以解决MTSP的NSGA-II代码文档

一、名词解释

1、染色体

MTSP的1个解决方案的搭载体。下文提到的染色体，其实是指解决MTSP具体问题的1个方案。在解决SD-MTSP(单仓库多旅行商问题，Single Depot Multiple Travelling Salesman Problem)时，我将染色体的组成重新定义。定义如下：

以8座城市，3个旅行商(n = 8, m = 3)为例：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 3 | 4 | 1 | 2 | 7 | 6 | 2 | 4 |

图1.1 搭载一个解的染色体

8座城市分别标记为:0,1,…,7;同理，n座城市分别标记为:0,1,…,n-1

染色体中，前n-1位为城市1～7的一个无序随机排列，后m-1位为分割位，为前7位的随机2座城市在染色体中的序号。其中，2指的是城市4在染色体中的序号，即chromosome[2] = 4，也指第1位旅行商所走路线为0->5->3->4->0。

整条染色体长度为：n+m-2，染色体前段长度为：n-1，后段长度为m-1。

染色体前段序列为chromosome[0]～chromosome[n-2]，后端序列为chromosome[n-1]～chromosome[n+m-3]。

染色体前段可取值为{1,2,…,n-1}，染色体后段可取值为{0,1,…,n-3}。

要求染色体后段必须是有序排列。

综上所述，一条染色体就这样搭载了一个解决方案。

2、种群

由多个染色体组成的集合，即MTSP有限解空间的子集。

3、适应度（函数）

在此题目的背景下，适应度由两项指标衡量，一是总路程指标，二是平衡系数指标。总路程即为一个解决方案中每个旅行商走过的路程之和，平衡系数即为在同一个解决方案中，各个旅行商所走路程的最大值减最小值。我习惯称之为用来衡量个体优良程度的性状。

而适应度函数就是用来计算单个个体的这两个性状。在代码中体现为总路程算子f1(X)，和平衡系数算子f2(X)，其中X表示染色体集合。

4、变异、交叉

根据遗传算法(GA)的基础知识可知，变异和交叉是GA中产生下一代种群的手段。在解决SD-MTSP时，我对变异和交叉的行为进行了重新定义。定义如下：

（1）变异

对于图1.1中的染色体来说，所有位点都可以变异，但为了防止变异出现畸形染色体，我对前段与后段的变异做出了区分。具体如下：

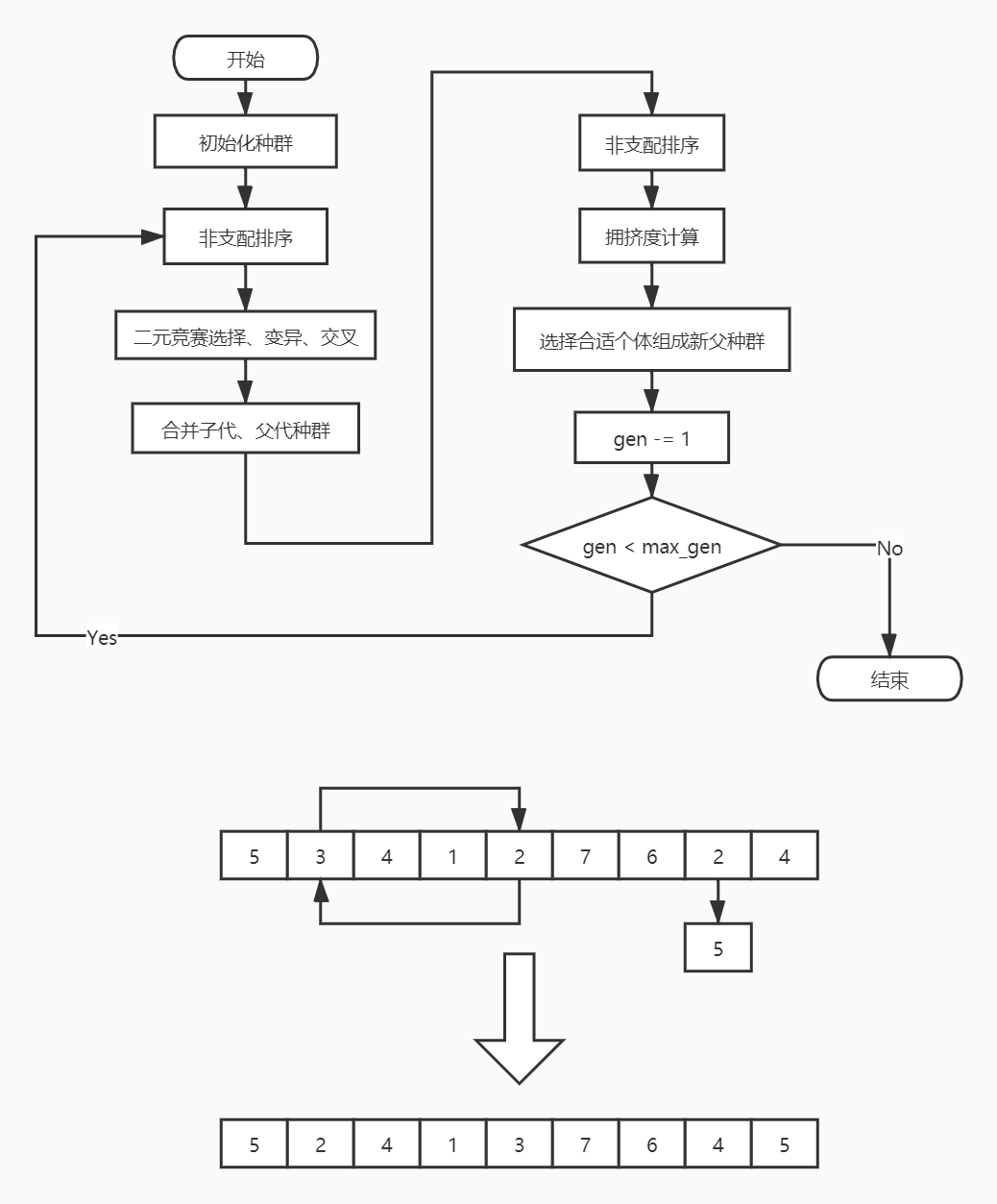


图1.2 染色体变异

由于前段是一个完整的城市随机序列，所以我将前段变异定义为随机选择前段两个位点进行交换，所以出现了图1.2的情况。并且我对染色体采取的定义是借鉴了前人提出的两部分染色体方法，该方法决定了染色体前段与后段是相互独立的。所以对于后段是可以进行单位点基因突变的。当然，为了保证染色体后段的有序，如果出现图1.2的情况需要对后段在变异后进行排序处理。

综上所述，方案1，**|**0->5->3->4->0**|**0->1->2->0**|**0->7->6->0**|**通过变异，就变成了方案2，**|**0->5->2->4->1->3->0**|**0->7->0**|**0->6->0**|**。

但是如同现实世界一样，变异所产生的染色体可能是畸形的。为了保证染色体经过变异依旧是正常染色体，我制定了以下规则：

1. 前段染色体变异只要按照交换方法进行就没有畸变的可能。
2. 为了保证后段染色体正常变异，首先根据原后段所有位点的基因，计算其对全集U(U={0,1,…,n-3})的补集，则得到的补集即为变异有效集。在生成的补集随机选取一个作为突变后的基因，最后对突变后的染色体后段进行排序。

（2）交叉

由交叉的一般概念得知，交叉是两条染色体的同一位置相互交换而导致染色体变异。同样，为了给交叉一个更加细致的定义，并且防止染色体畸变，我做出了如下定义：

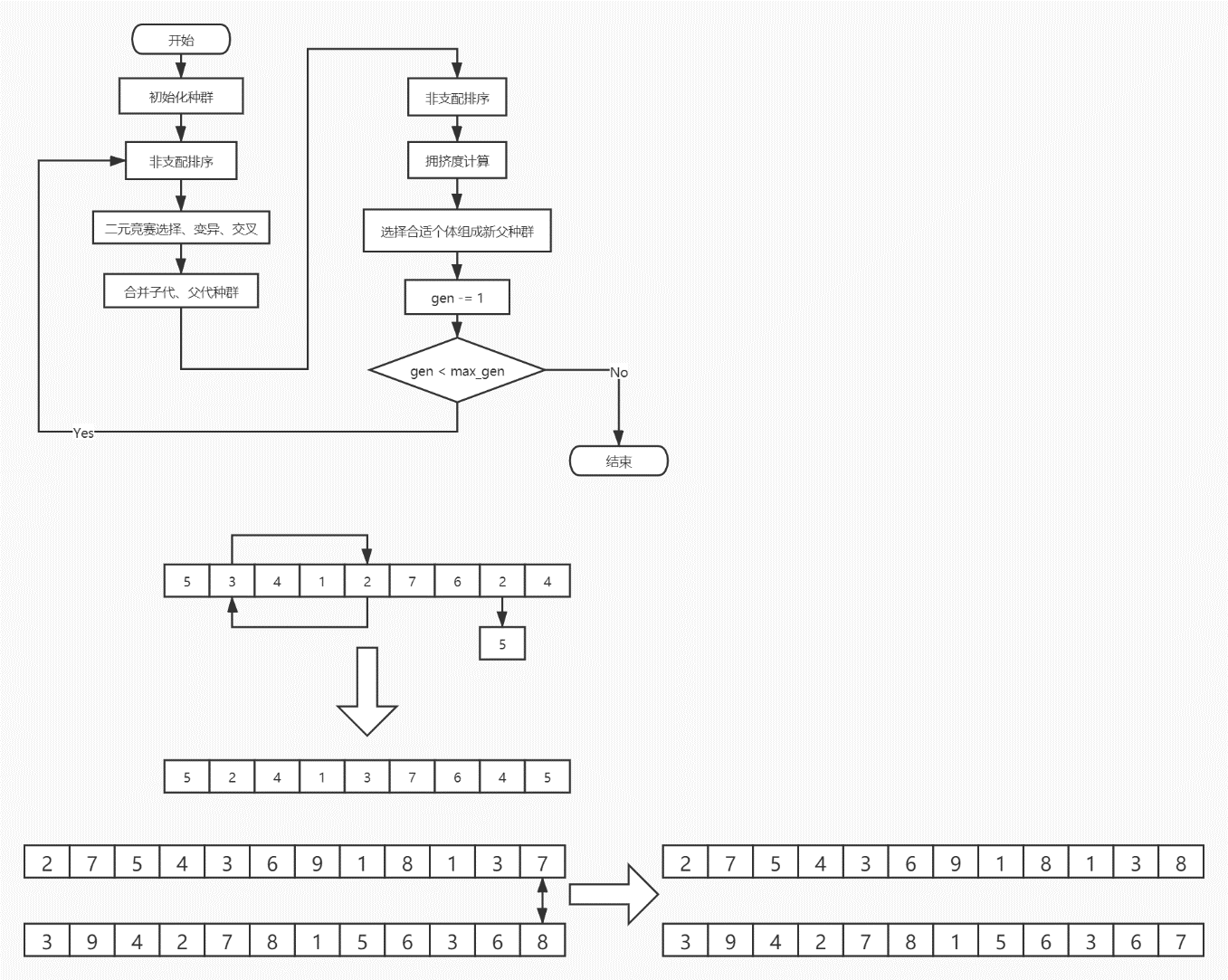


图1.3 染色体交叉

首先，由于染色体前段的完整性，如果对染色体前段进行交叉就一定会造成染色体畸变，所以不对染色体前段进行交叉操作，只对染色体后段进行交叉操作。染色体后段交叉的一般流程为，随机选择一个位点，检测位点是否可以交叉，可以则进行交叉操作。

检测的具体步骤定义为如下：

1. 考察染色体1与染色体2的后段是否有交集，如果有，则记录交集；
2. 通过遍历两个染色体后段序列找到交集中的无效交叉位点的位置，在对其进行求补集操作，得到两个染色体可以进行交叉的位点；
3. 进行交叉操作，并对交叉后的染色体后段排序，保证其有序性。

5、二元选择竞赛(Binary Tournament Selection)

二元选择竞赛是在父代种群繁衍子代（假子代，unshaped offspring）种群时进行的基于精英策略的择优操作。

在本命题下，二元竞赛选择根据总路程和平衡系数两个指标，每次选取N/2个样本进行择优，选出两个较优的个体，随之进行突变、交叉，得到假子代的两个个体。由于突变和交叉均由概率指导，所以假子代中是可以出现父代染色体未经变异直接留下的染色体。

二、程序流程图

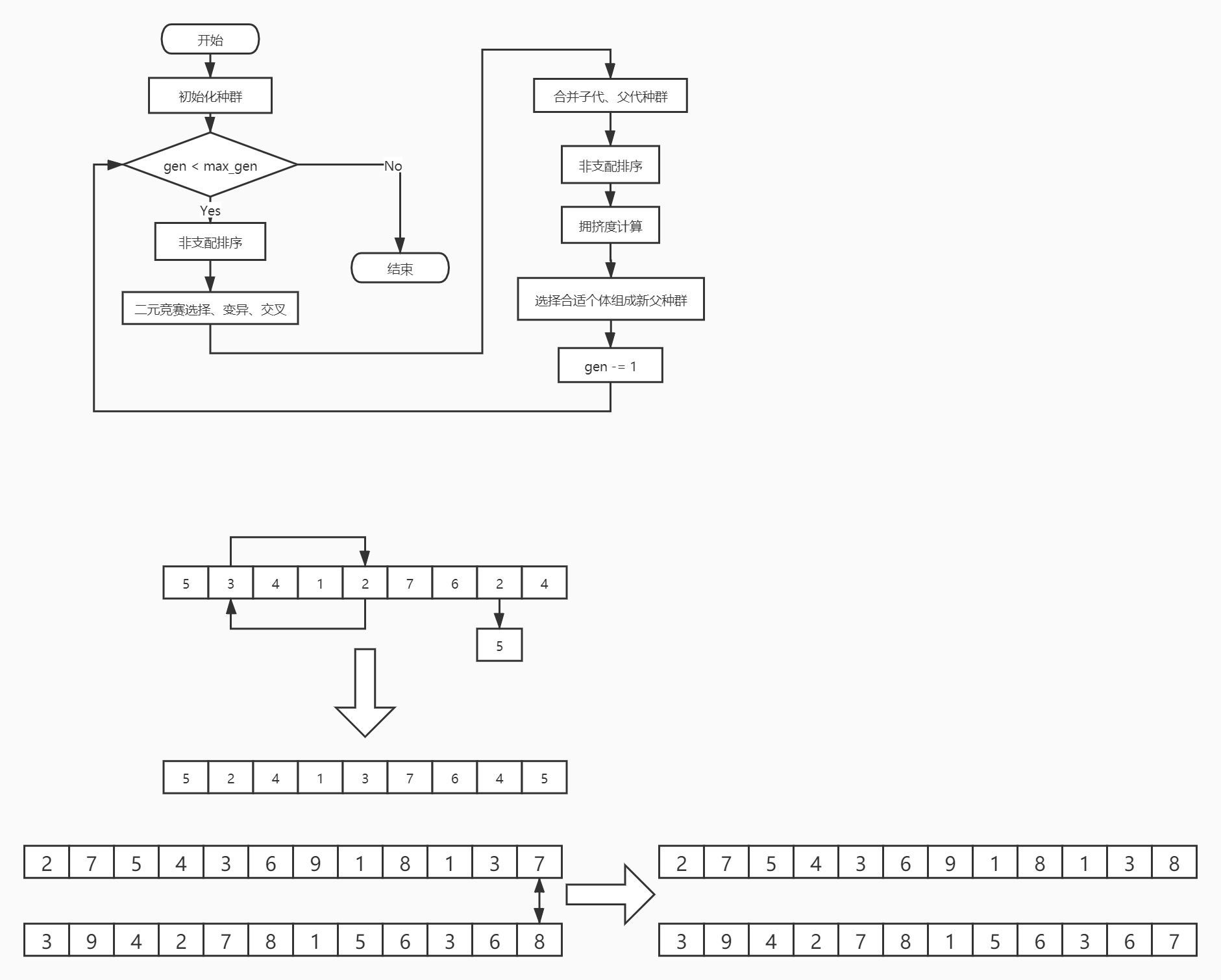


图1.4 NSGA-II总流程图

图1.4为第一版解决SD-MTSP的NSGA-II算法流程。整个流程从生物进化角度可以看成种群迭代的过程，种群从初始状态通过计算对应目标函数，得到每个染色体的适应度，通过依据非支配的定义和f1(X)、f2(X)进行非支配排序，赋予每个染色体对应等级，再通过二元竞赛选择对每次选出的两个染色体进行突变和交叉得到假子代的两个个体，循环N/2（N为染色体个数）次后，合并父代子代，在次计算相应参数和等级，通过等级筛选不同等级下更优者，通过拥挤度筛选同等级下更优者，最后得到新父代种群，若种群代数未超过设定最大迭代次数，则重复上述过程。

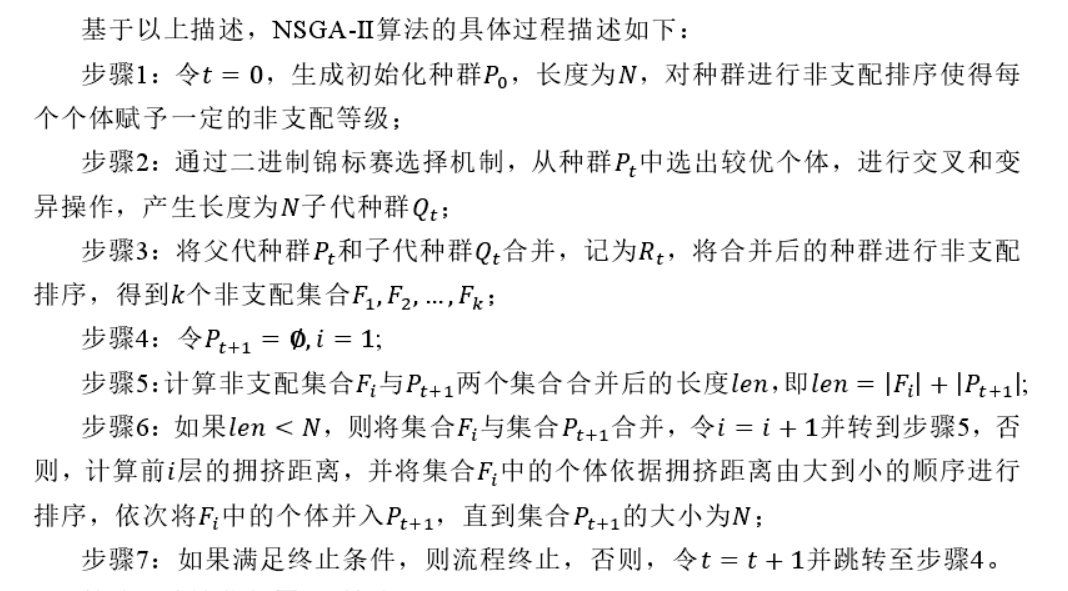


图1.5 NSGA-II算法具体过程

图1.5参考自：杨帅. 求解多旅行商问题的进化多目标优化和决策算法研究[D].武汉科技大学,2020.

三、程序概述

1、类(class)

（1）城市类

类属性：

城市个体集：individuals[]

距离矩阵：distance\_matrix[]

城市数量：n

类方法：

计算距离矩阵：cal\_distance\_matrix() #距离矩阵为上三角矩阵

查找两座城市之间的距离：x\_y\_find\_i(c1, c2)

（2）种群类

类属性：

size：种群大小

n：城市数

m：旅行商数

mutation\_pro：变异概率

frontm\_pro：染色体前段变异概率，后段变异概率为1 - frontm\_pro

crossover\_pro：交叉概率

max\_gen：最大迭代次数

gen：迭代次数

generation：种群的代数，即迭代次数

individuals：个体集合

total\_d：个体总路程集合，f1(X)，X代指染色体集合

balance\_factor：平衡系数，f2(X)，X代指染色体集合

d：每个染色体中每个旅行商的路程集合

np：支配计数

sp：该个体所支配的其他个体集合

rank\_list：非支配排序后的集合

rank：非支配排序后每个染色体对应的等级

unsQ：unshaped offspring, 即假子代，为二重列表

crowding\_distance：拥挤度列表

crowding\_sorted\_list：经过拥挤度排序后的列表

类方法：

总路程算子：f1(city, chrom)

平衡系数算子：f2()

非支配排序算子：fast\_nondominated\_sort(chrom\_size)

产生假子代：unshaped\_evolution()

交叉：crossover(chrom1, chrom2)

突变：mutation(chrom)

拥挤度算子：crowding\_distance\_computation(r\_list, miss\_num)

产生新父种群：produce\_newf(city)

迭代入口：run(city)

工具类：

集合取反：produce\_mutation\_set(num, list) #产生有效变异集合

计算交集，并返回可突变集合：whether\_intersect(list1, list2)

快速排序：QuickSort(list, low, high)

快排中的划分：partition(list, low, high)

三、遗留问题

1、代码优化

尽管最开始去看了Hairs Ali Khan在github上NSGA-II的代码，但是最终决定自己亲自手写来理解NSGA-II，实践证明理解相对更透彻一点。但是由于个人的代码水平和对python的理解应用有限，所以在我个人看来代码需要优化，但对于具体如何优化没有个清晰的概念，只是觉得自己的代码可能存在逻辑冗余、长度过长等缺陷，部分代码理解起来较绕，不好理解和理清。

2、交叉、突变优化

由于我在网上并没有查到具体的交叉、突变方式，论文中又没有详细的逻辑描述，找到的代码是以数学计算为例，所以我自己花了些时间去定义了交叉和突变方法，对于目前仅以生成随机数并进行覆盖的形式是否有更好的方法和优化，我没有一个很好的概念和把握，且个人数学功底不是很扎实和过硬，无法进行猜想和证明。

3、参数优化

现已有参数有：

* 染色体数目N
* 城市数目n
* 旅行商数目m
* 突变概率m\_pro
* 交叉概率c\_pro
* 最大迭代次数max\_gen
* 染色体前段突变概率fm\_pro
* 城市坐标

（1）城市数目、旅行商数目、城市坐标

这三者其实是问题的已知，是需要用户输入的。

但是是否需要对用户输入的数据先进行一个数学上的检测呢？用来判断输入是否合法，不会引起程序的崩溃；判断是否相对合理，是否可以计算出解空间向最优解逼近的程度；是否可以计算出一个染色体数目的大致范围，使之尽量不要陷进局部最优或者类似于“过饱和”的情况等。

（2）染色体数目、最大迭代次数

这两者是保证解集尽量向最优解逼近的第一道防线，对于这两者的控制如果控制得好的话可以尽量减少向局部收敛或类“过饱和”的可能。

对于同一组城市坐标，如下：

城市数据：（N=8，n=20，m=4）

[34, 53], [96, 62], [29, 21], [10, 48], [14, 3], [3, 64], [13, 77], [48, 13], [69, 53], [79, 13], [26, 33], [85, 43],[63, 56], [2, 57], [69, 93], [79, 31], [35, 35], [26, 96], [60, 47], [30, 74]

第0代个体总路程(f1)：

[1135, 1177, 888, 1111, 1095, 1104, 1118, 1110]

第0代个体平衡系数(f2)：

[343, 98, 316, 296, 618, 340, 423, 725]

第100代个体总路程(f1)：

[763, 761, 762, 760, 763, 737, 750, 728]

第100代个体平衡系数(f2)：

[55, 65, 60, 66, 55, 132, 76, 132]

第100代个体总路程(f1)：

[747, 736, 809, 823, 758, 759, 808, 849]

第100代个体平衡系数(f2)：

[30, 306, 53, 44, 257, 56, 55, 94]

第200代个体总路程(f1)：

[756, 756, 756, 734, 737, 756, 739, 731]

第200代个体平衡系数(f2)：

[39, 39, 39, 186, 185, 39, 90, 255]

第1000代个体总路程(f1)：

[622, 622, 622, 622, 622, 622, 622, 637]

第1000代个体平衡系数(f2)：

[164, 164, 164, 164, 164, 164, 164, 164]

第500代个体总路程(f1)：

[728, 728, 721, 663, 637, 606, 665, 687]

第500代个体平衡系数(f2)：

[77, 77, 81, 114, 153, 230, 99, 113]

第150代个体总路程(f1)：

[744, 735, 745, 755, 763, 804, 782, 733]

第150代个体平衡系数(f2)：

[60, 60, 60, 60, 60, 14, 51, 60]

第1000代个体总路程(f1)：

[653, 653, 653, 653, 653, 653, 653, 640]

第1000代个体平衡系数(f2)：

[99, 99, 99, 99, 99, 99, 99, 99]

第1000代个体总路程(f1)：

[610, 610, 610, 610, 604, 599, 613, 590]

第1000代个体平衡系数(f2)：

[131, 131, 131, 131, 172, 173, 114, 211]

（3）突变概率、交叉概率和染色体前段变异概率

染色体后段变异概率 = 1 – 染色体前段变异概率

这三者是第二道防线，但同时也在对种群的冗余程度和收敛速度起到一定的影响。

综上，参数优化是重中之重，对于处理冗余问题、收敛速度问题等有着很重要的地位，而如何设计和实现通过数学手段来获得参数优化的相对好的办法和范围条件等是关键。