NSGA2算法解决**ZDT**系列问题

1. 总结**:**概述所考虑的多目标目标优化算法

在单目标优化问题中，通常最优解只有一个。而在多目标优化问题中，各个目标之间相互制约，可能使得一个目标性能的改善往往是以损失其他目标性能为代价，不可能存在一个使所有目标性能都达到最优的解（这就意味着这两个目标可能存在较大的负相关性），所以对于多目标优化问题，其解通常是一个非劣解的集合–帕累托解集。我们需要找到一组尽可能接近帕累托最优域的，尽可能不同的解。

NSGA2算法提出了快速非支配的排序算法，降低了计算非支配序的复杂度，使得优化算法的复杂度由原来的 降为 （ 为目标函数的个数， 为种群的大小）。引入了精英策略，扩大了采样空间。将父代种群与其产生的子代种群组合在一起，共同通过竞争来产生下一代种群，这有利于是父代中的优良个体得以保持，保证那些优良的个体在进化过程中不被丢弃，从而提高优化结果的准确度。并且通过对种群所有个体分层存放，使得最佳个体不会丢失，能够迅速提高种群水平。引入拥挤度和拥挤度比较算子，这不但克服了NSGA算法中需要人为指定共享参数 的缺陷，而且将拥挤度作为种群中个体之间的比较准则，使得准Pareto域中的种群个体能均匀扩展到整个Pareto域，从而保证了种群的多样性。

2. 描述**:**详述在适应度分配、多样性维护涉及的其他成分的背景下的算法设计

在NSGA2算法中，主要是对于NSGA算法的改进，在以下方面有了提升。

1. 快速非支配排序

可以由原来的 $O(MN^3)$ 降低到 $O(MN^2)$。实现的方式就是维护两个变量：支配个数 N_p，被支配个体集合 S_p。

伪代码：

```
def fastNondominatedSort(P): # 初始化群体为P
    F = [ ]
    for p in P: # 遍历
        Sp = [ ] # 设置被支配个体的集合
         np = 0
         for q in P:
             if p > q: #如果p支配q，把q添加到Sp列表中
                 Sp.append(q)
             else if p < q: #如果p被q支配，则把np加1
                 np += 1
        if np == 0:
```

```
            p_rank = 1          #如果该个体的np为0，则该个体为Pareto第一级
        F1.append( p )
    F.append( F1 )
    i = 0
    while F[i]:
        Q = [ ]
        for p in F[i]:
            for q in Sp: #对所有在Sp集合中的个体进行排序
                nq -= 1
                if nq == 0: #如果该个体的支配个数为0，则该个体是非支配个体
                    q_rank = i+2  #该个体Pareto级别为当前最高级别加1。此时i初始值为0，所以要加
2

                    Q.append( q )
        F.append( Q )
        i += 1
```

2. 精英主义

算法步骤：

1. 首先将父代种群 P_i 和子代种群 D_i 合成种群 R_i。
2. 根据以下规则从种群 R_i 中生成新的父代种群 C_{i+1}：

    ①根据Pareto等级从低到高的顺序，将整层种群放入父代种群 C_{i+1} ，直到某一层该层个体不能全部放入父代种群

    ②将该层个体根据拥挤度从大到小排列，依次放入父代种群 C_{i+1}，直到父代种群 C_{i+1} 填满

3. 拥挤度

为了使得到的解在目标空间中更加均匀，引入了拥挤度的概念，在二维空间的里面，可以理解该个体在目标空间所能生成的最大的矩形（该矩形不能触碰目标空间其他的点）的边长之和。

算法步骤（伪代码）：

1. 设置拥挤度变量为 x_d，$x \in [1, n]$
2. for each fm:
    1. 根据该目标函数对该等级的个体进行排序，记录 f_m^{max}，f_m^{min} 为最大值和最小值
    2. 对于排序后两个边界的拥挤度设置为正无穷大
    3. 计算迭代值：$t = (f_m(i+1) - f_m(i-1))/(f_m^{max} - f_m^{min})$
    4. 更新拥挤度：$x_d = x_d + t$

4. 多项式变异与模拟二进制交叉

针对使用二进制编码的单点交叉具有的Average Property 和 Spread Factor Property ，我们使用概率密度函数的方式进行模拟。

首先根据公式：（1） p1+p2=c1+c2 （2）factor = |(c1-c2)/(p1-p2)| 我们可以解出 c1和c2

c1=(1/2)*f*(p2+p1)-(1/2)*f*(p2-p1) c2=(1/2)*f*(p2+p1)+(1/2)*f*(p2-p1)

因此，只要生成不同的 factor 我们就可以得到不同的 (c1,c2) 的解。

我们采用概率密度函数来拟合 factor

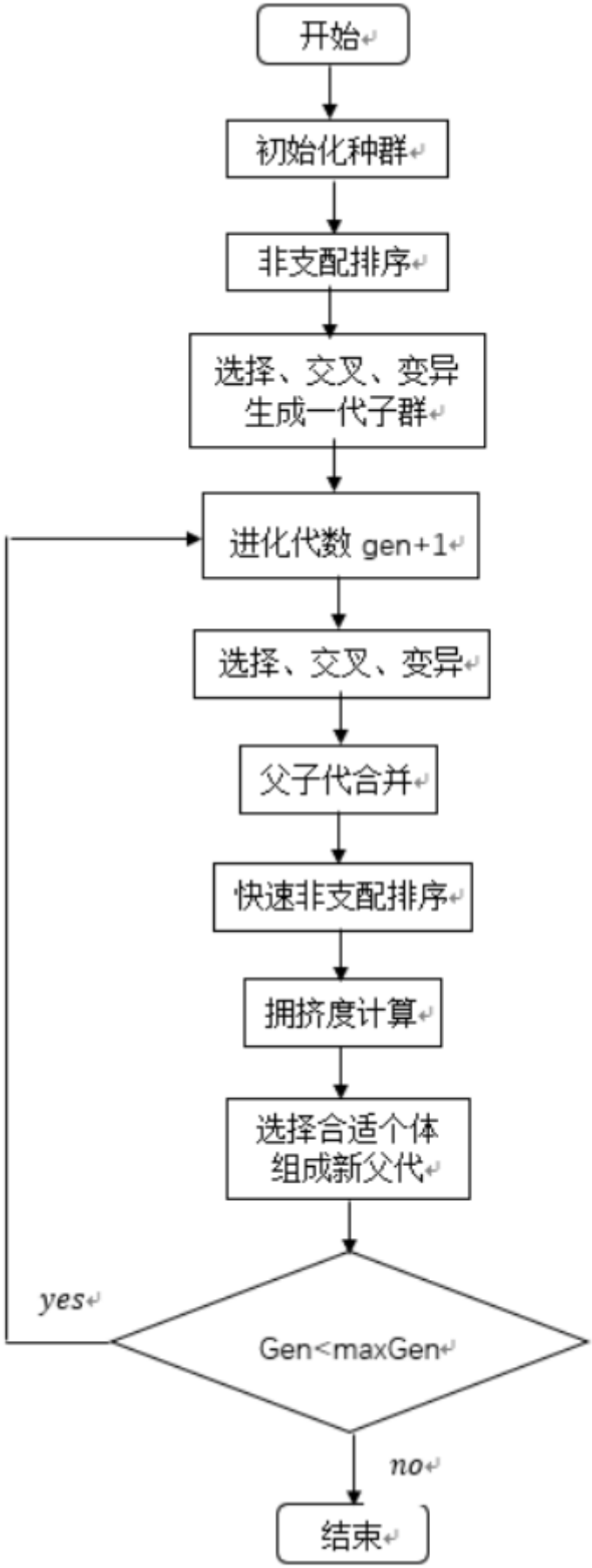
根据之前推导的公式1和2，可以得到child1和child2

5. 锦标赛算法

用锦标赛算法可以更好的去找最大和第二大的数。

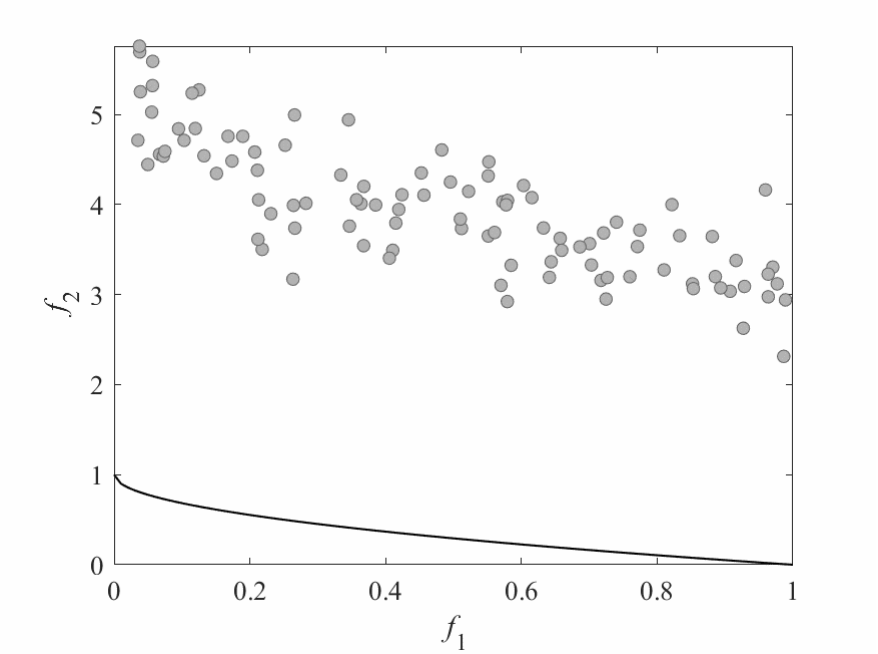
```
public static int ChampionShip(int a[]) {
    int temp[] = new int[2*a.length]
    int secondmax = -0x3f3f3f3f

    // 将数组里面的数据填在临时数组中的temp[a.length-1]的后面
    for(int i=temp.length-1;i>=a.length;i--) temp[i] = a[i-a.length]

    // 从后往前进行比较，将最大值以此填在temp[1]到temp[a.length-1]之间，则temp[1]是数组中的最大值
    for(int i=temp.length-2;i>=2;i-=2) temp[i/2]=Math.max(temp[i], temp[i+1]);

    // 我们将从直接输给最大值的数中找到第二大的数
    for(int i=1;i<a.length;) {
        if(temp[i]==temp[2*i]) {
            if(secondmax<temp[2*i+1]) secondmax=temp[2*i+1];
            i=i*2;
        }
        else{
            if(secondmax<temp[2*i]) secondmax=temp[2*i];
            i=2*i+1;
        }
    }
    return secondmax
}
```

算法总流程

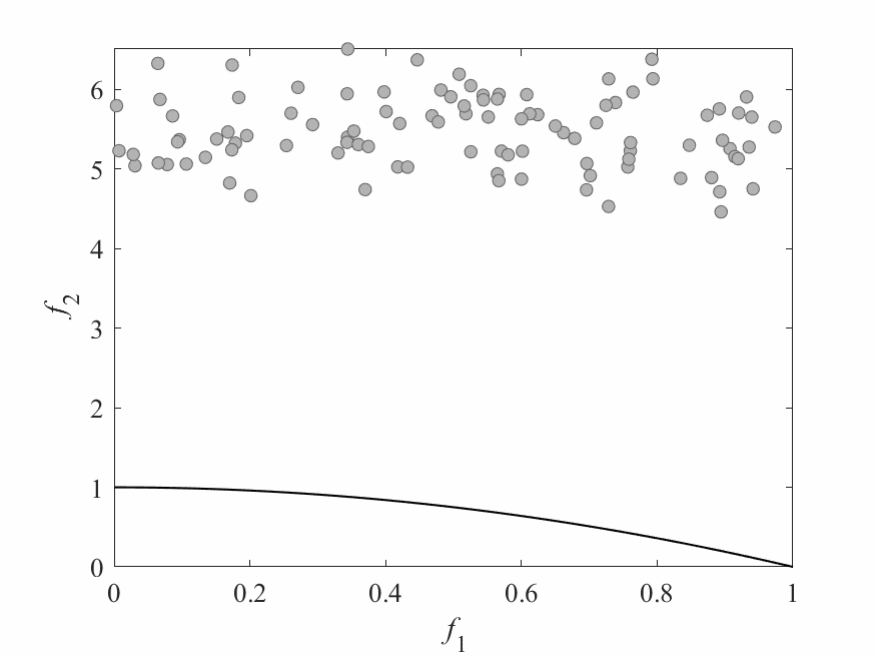

开始
初始化种群
非支配排序
选择、交叉、变异
生成一代子群
进化代数 gen+1
选择、交叉、变异
父子代合并
快速非支配排序
拥挤度计算
选择合适个体
组成新父代
yes
Gen<maxGen
no
结束

3. 模拟**:**展示模拟结果**(**非支配前沿**)**和统计结果

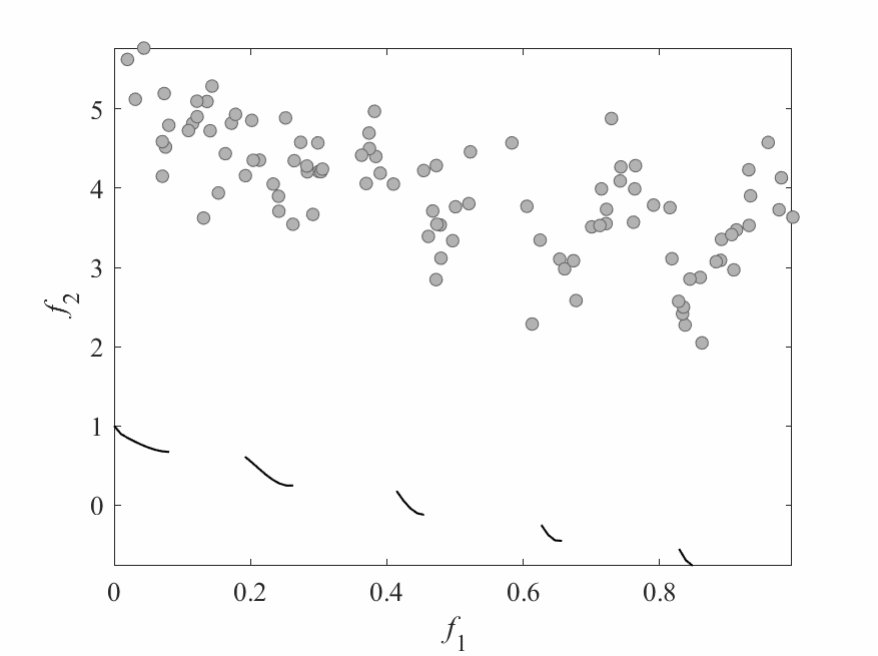
ZDT1



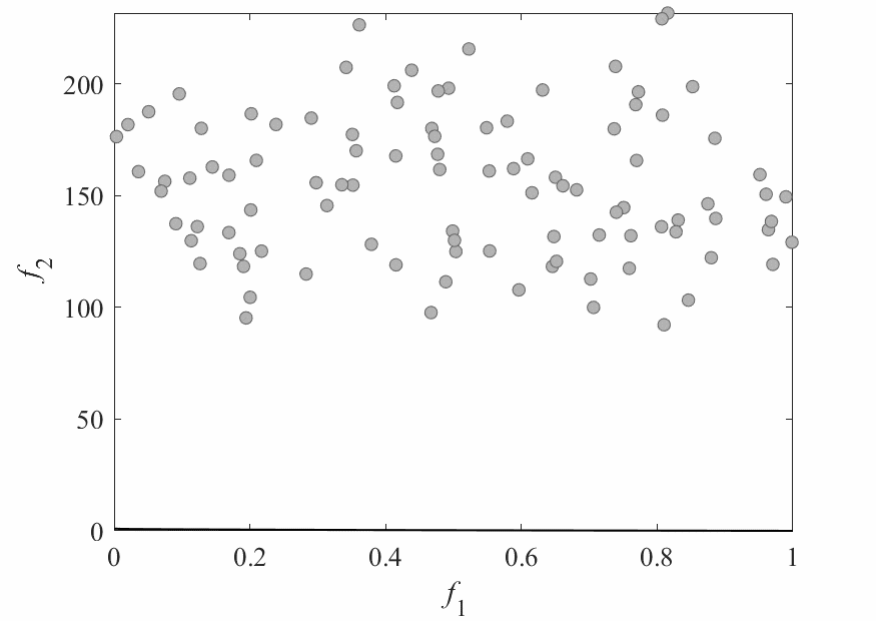
ZDT2



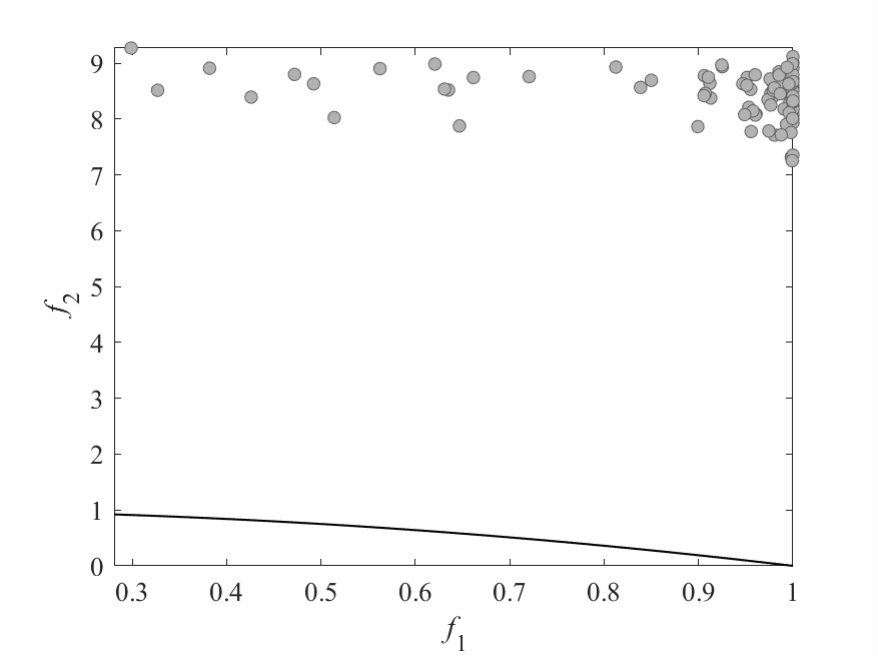
ZDT3



ZDT4



ZDT6



4. 观察**:**记录算法的优缺点

1. 优点

    1. 在第一代的基础上提出了快速非支配排序算法，一方面降低了计算的复杂度，另一方面它将父代种群跟子代种群进行合并，使得下一代的种群从双倍的空间中进行选取，从而保留了最为优秀的所有个体；
    2. 引进精英策略，保证某些优良的种群个体在进化过程中不会被丢弃，从而提高了优化结果的精度；
    3. 采用拥挤度和拥挤度比较算子，不但克服了 $NSGA$ 中需要人为指定共享参数的缺陷，而且将其作为种群中个体间的比较标准，使得准 $Pareto$ 域中的个体能均匀地扩展到整个 $Pareto$ 域，保证了种群的多样性。

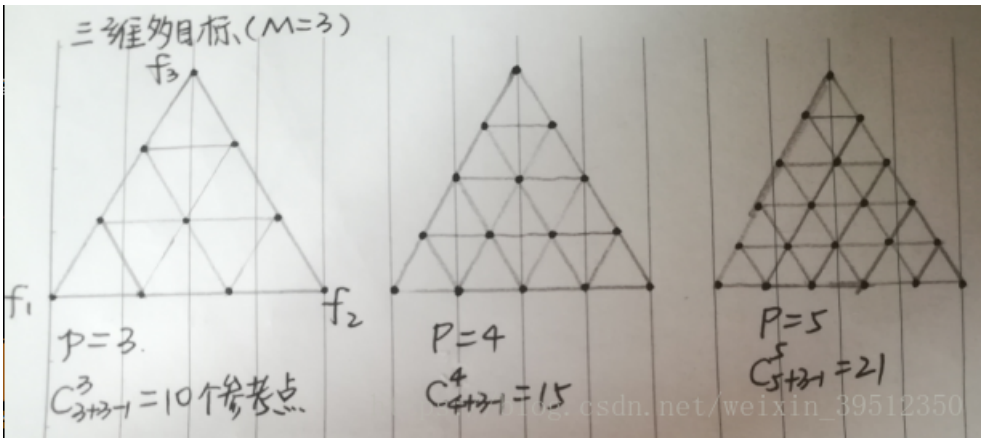
2. 缺点

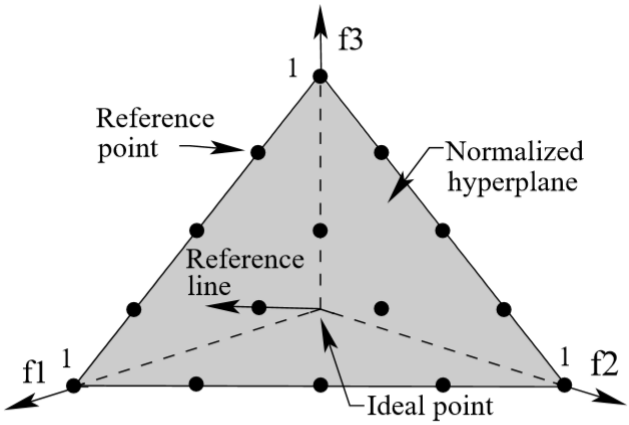
    1. 随着优化目标数量的增加，非支配解在种群中的比例也在增加，因而会导致搜索过程缓慢；
    2. 对于高维目标空间，维持多样性的指标计算复杂度过高，解的邻近元素寻找困难；
    3. 对于高维目标空间，重组算子的搜索能力过于低效了；
    4. 当面对三个及以上的多目标优化问题时，如果继续采用拥挤距离的话，$NSGA2$ 算法的收敛性和多样性会不好，容易陷入局部最优。

5. 潜在改进**:**根据你在问题**2**中的设计，提供你的想法来推进提议的概念

- 参考点：出自 1998 年 Das 和 $Dennis$ 在1998年提出的边界交叉构造权重的方法。

- 构造参考点：在标准化参考平面上，维数是 m， 如果每一维目标被均匀地分割成 p 份，那么会均匀地产生 C_{m+p-1}^{p} 个参考点。





- 引入参考点的目的：就是为了获取种群个体与响应参考点之间的映射关系（即垂直距离），进而使得种群超更接近参考点的方向进化，进而事参考点的分布更加均匀。

- 规范化目标函数：

  - 以最小化为例，首先得到种群 S_t 中的所有个体在每一维目标上的最小值，构成当前种群的理想点。
  - 将所有个体的目标值，以及理想点以此理想点为参考作转换操作，这时理想点变为原点，个体的目标值为转换后的临时标准化目标值。
  - 然后计算每一维目标轴上的极值点，这 M 个极值点组成了 $M-1$ 维的线性超平面，这时可以计算出各个目标方向上的截距。然后利用截距和临时的标准化目标值计算真正的标准化目标值。
  - 计算公式：$f_i^n(x)=\frac{f_i(x)-z_i^{min}}{a_i-z_i^{min}}, i\in[1,m]$ ，其中 a_i 是截距。

- 关联操作：在设置完参考点之后，要进行关联操作，我们要让种群中的个体分别关联到相应的参考点。为了这个目的，我们定义一个参考线，它是原点与参考点在目标空间的连线。有了参考线后，我们计算种群 S_t 的每个个体到参考线的垂直距离。然后个体与和它最近的参考线关联起来。

6. 结论**:**得出相关结论

在ZDT系列的问题上有很好的拟合程度

7. 附录**:**代码

```
确定种群大小 n，交叉概率 t，迭代次数 g
随机产生 n 个个体，它们整体视为种群 P
for i = 1 to g
      P = Ø
      for j = 1 to n
            产生一个 [0,1] 的随机数 a
            if (a<t)
                  从 P 中随机选出两个个体作为父母，交叉产生一
                  个新的个体并放入 P’ 中
            else
                  从 P 中随机选出一个个体，变异产生一个新的个
                  体并放入 P’ 中
            end
      end
      利用非支配排序和拥挤距离，从 P∪P’ 中选出 n 个个体，      代替 P           # NSGA-II改进
的部分
end
输出最终种群 P 中的非支配个体
```