Optimization Methods

👉 常用的最优化方法

- 线性规划
- 整数规划
- 非线性规划
- 动态规划
- 多目标规划
- 对策论？？？

基本概念

一元函数极值基础

👉 驻点——导数为0的点。

👉 奇点——使函数连续，但导数不存在的点

👉 极值问题——求某个函数的极值，但其中的变量**收到一些条件的约束**。有约束的极值问题称为**条件极值问题**，无条件的极值问题称为**无条件极值问题**

如函数, $f(x,y)$, 在条件, $\varphi(x,y)=0$, 下的极值问题——, $f(x,y)$, 是目标函数，, $\varphi(x,y)=0$, 是约束条件

👉 等式约束下的条件极值问题的数学模型——拉格朗日乘数法

$$min_x f(x)\ s.t\quad g_i(x)=0, i=1,2,\cdots,m$$

求函数, $f(x,y)$, 在条件, $\varphi(x,y)=0$, 下的极值，运用拉格朗日乘数法

$$构造函数, F(x,y,\lambda)=f(x,y)+\lambda\varphi(x,y)\ 令 \nabla F=0 \Rightarrow 解出 (x_0,y_0)\text{，则该点就是可能的极值点}$$

范数

向量范数

☞ 常用范数

- 1范数

$$||x||_1 = |x_1| + |x_2| + \cdots + |x_n|$$

- 2范数

$$||x||_2 = (|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2)^{\frac{1}{2}} = (x^T x)^{\frac{1}{2}}$$

- 无穷范数

$$ ||x||*infty = max*{1\leq i \leq n} |x_i| $$

☞ 范数性质

- Holder不等式

$$|x^T y| \leq ||x||_p ||y||_q, \frac{1}{p} + \frac{1}{q} = 1 \text{ 特别的，} p = q = 2\text{时，是柯西}-\text{施瓦茨不等式}|x^T y| \leq ||x||_2 ||y||_2$$

矩阵范数

☞ 常用的矩阵范数

$$ A\in P^{m\times n}\ ||A||*{m_1} = \sum*{j=1}^{n}\sum_{i=1}{m} |a_{ij}|\ ||A||*{m_2} = (\sum*{j=1}^{n}\sum_{i=1}{m} |a_{ij}|^2)^{\frac{1}{2}}\ ||A||*{m*\infty} = max_{i,j} {|a_{ij}|}\ ||A||*F = (\sum*{j=1}^{n}\sum_{i=1}{n} |a_{ij}|^2)^{\frac{1}{2}}\quad F范数 $$

梯度

$$\text{函数}f(x)\text{存在一阶偏导数}, x \in R^n, \text{称向量 } \nabla f(x) = (\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_n})^T \text{ 为}f(x)\text{在点}x\text{处的梯度}$$

☞ 梯度方向，函数值增长最快；负梯度方向，函数值增长最慢

☞ 常用的梯度

$$\text{常数函数}c \Rightarrow \nabla c = (0, 0, \cdots, 0)^T = 0$$

$$\text{对}b = (b_1, b_2, \cdots, b_n)^T \in R^n, x = (x_1, x_2, \cdots, x_n)^T\ b^T x = b_1 x_1 + b_2 x_2 + \cdots + b_n x_n\ \nabla(b^T x) = b$$

$$\nabla(x^T x) = \nabla(x_1^2 + x_2^2 + \cdots + x_n^2) = 2x$$

$$\text{若}A\text{是对称矩阵，则 }\nabla(x^T A x) = 2Ax$$

向量值函数的导数

☞ 向量值函数

$$g(x) = (g_1(x), g_2(x), \cdots, g_m(x))^T, \text{其中}x = (x_1, x_2, \cdots, x_n)^T$$

☞ 向量值函数的一阶导数(Jacobi矩阵)

$$\nabla g(x) = \begin{pmatrix} \frac{\partial g_1(x)}{\partial x_1} & \frac{\partial g_1(x)}{\partial x_2} & \cdots & \frac{\partial g_1(x)}{\partial x_n} \\ \frac{\partial g_2(x)}{\partial x_1} & \frac{\partial g_2(x)}{\partial x_2} & \cdots & \frac{\partial g_2(x)}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial g_m(x)}{\partial x_1} & \frac{\partial g_m(x)}{\partial x_2} & \cdots & \frac{\partial g_m(x)}{\partial x_n} \end{pmatrix}$$

☞ 向量值函数的二阶导数(Hesse矩阵)

$$\nabla^2 f(x)=\begin{pmatrix}\frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1\partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1\partial x_n}\\ \frac{\partial^2 f(x)}{\partial x_2\partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2\partial x_n}\\ \cdots & \cdots & \cdots\cdots & \\ \frac{\partial^2 f(x)}{\partial x_n\partial x_1} & \frac{\partial^2 f(x)}{\partial x_n\partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2}\end{pmatrix}$$

凸集和凸函数

👉 是非线性规划的基本概念

👉 凸集

定义8.3 设$S\in R^n$，若对任意$x^{(1)},x^{(2)}\in S$，及$\lambda\in[0,1]$，都有$\lambda x^{(1)}+(1-\lambda)x^{(2)}\in S$，则称$S$为**凸集**. $\lambda x^{(1)}+(1-\lambda)x^{(2)}$称为$x^{(1)}$和$x^{(2)}$的**凸组合**.

👉 凸函数

定义8.4 设$f:\quad S\subset R^n\to R^1$，$S$是**凸集**，如果对所有的$x^{(1)},x^{(2)}\in S$，$\lambda\in(0,1)$，都有

$$f(\lambda x^{(1)}+(1-\lambda)x^{(2)})\le\lambda f(x^{(1)})+(1-\lambda)f(x^{(2)}),$$

则称$f(x)$为S上的**凸函数**.

凸函数的**线性运算(数乘和加法)**(但是数乘的系数必须非负，线性组合也必须非负)之后还是凸函数

凸函数的重要性

定理8.6 设 S为R^n中的一个非空凸集，$f(x)$是定义在S上的凸函数，则$f(x)$在S上的局部极小点是**全局极小点**，且极小点的集合为**凸集**.

凸函数的一阶判别条件

定理8.7 设$f:\quad S\subset R^n\to R^1$，$S$是非空凸集，且$f(x)$在$S$上可微，则$f(x)$是在$S$上的**凸函数**的充要条件是对任意$x^{(1)},x^{(2)}\in S$，成立

$$f(x^{(2)})\ge f(x^{(1)})+\nabla f(x^{(1)})^T(x^{(2)}-x^{(1)}).$$

凸函数的二阶判别条件

定理8.8 设$f: \quad S \subset R^n \to R^1$，$S$是非空凸集，且$f(x)$是$S$上的二次可微函数，则$f(x)$是在$S$上的凸函数的充要条件为

对$\forall x \in S$，在x处的Hesse矩阵是半正定的.

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \cdots \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}$$

👉 凸规划

$$ 设f(x)是凸函数，g(x)是凹函数，h_j(x)是线性函数，下面的规划是凸规划\ min \quad f(x) \quad x\in R^n\ s.t \quad g_i(x) \geq 0 \quad i = 1,\cdots,m\ h_j(x) =0 \quad j = 1,\cdots,l

$$

线性规划

👉 化为标准形式

👉 单纯形法

👉 Matlab求解——`linprog`

以下图为例

$$\min F(x) = 0.03x_1 + 0.22x_2 + 0.38x_3 + 0.5x_4 + 0.35x_5$$

$$s.t. \begin{cases} 0 \leq x_1 \leq 0.3 \\ 0 \leq x_2 \leq 0.25 \\ 0.1 \leq x_3 \leq 0.3 \\ 0.2 \leq x_4 \leq 1 \\ 0.05 \leq x_5 \leq 0.2 \\ x_1 + x_2 + x_3 + x_4 + x_5 = 1 \end{cases}$$

```
%目标函数中决策变量前的系数
f = [0.03 0.22 0.38 0.5 0.35];
lb = [0 0 0.1 0.2 0.05];%每一个决策变量的下界
ub = [0.3 0.25 0.3 1 0.2];%每一个决策变量的上界
A = []; b = [];
Aeq = [1 1 1 1 1]; beq = 1;
[optx,optvalue,flag] = linprog(f,A,b,Aeq,beq,lb,ub)
```

$$\min_x f^T x \text{ such that } \begin{cases} A \cdot x \le b, \\ Aeq \cdot x = beq, \\ lb \le x \le ub. \end{cases}$$

其中关于`A b Aeq beq`,

`linprog`是求最小值，若求最大值只需在目标函数添一个负号

若决策变量均是整数，则属于整数规划，可以在Matlab中用`intlinprog`，也可以用Lingo软件

$$\min f(x) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$
$$s.t. \begin{cases} x_2 + 2x_3 + 3x_4 + 5x_5 + 6x_6 \ge 2000, \\ 5x_1 + 4x_2 + 3x_3 + 2x_4 + x_5 \ge 1000, \\ x_i \ge 0 (i = 1, 2, \cdots, 6) \text{且为整数}. \end{cases}$$

用Lingo求解整数规划如下

```
x2 + 2*x3 + 3*x4 + 5*x5 + 6*x6 <= 2000; !约束条件
5*x1 + 4*x2 + 3*x3 + 2*x4 + x5 <= 1000;
f1 = x1+ x2 + x3 + x4 + x5 + x6;  !目标函数
min = f1;
x1>=0;x2>=0;x3>=0;x4>=0;x5>=0;x6>=0;
@GIN(x1);@GIN(x2);@GIN(x3);@GIN(x4);@GIN(x5);@GIN(x6);
END
```

非线性规划

☞ 定义：目标函数或约束条件中至少有一个是非线性函数的最优化问题

MC求解非线性规划

见**最优化模型——直接搜索法和MC法**这一节

最优化模型——直接搜索法和MC法

☞ 无约束，有约束都可以

☞ 对于👇的模型分别使用直接搜索和MC

$$min \quad f(x) \; s.t. \quad g(x) \le 0 \; l_i \le x_i \le u_i \quad i = 1, 2, \cdots, n$$

☞ 直接搜索法

要设置步长搜索

以二元函数为例

```
tmin = inf ;
for x1 = l1:s1:u1 %s1=(u1-l1)/N
    for x2 = l2:s2:u2 %s2 = (u2-l2)/N
        if f(x)<tmin and g(x) <= 0:
            tmin = f(x); %存储最优值
            tx   = x; %存储决策
        endif
    endfor
endfor
```

☞ MC法

- 无约束

```
N = 1e6; %投点次数
min_val = 1e5;
for i = 1:m
```

```
    for j = 1 : n %n是决策变量的维度
        r(j) = rand; %r(j)是在[0,1]均匀分布
        x(j) = l(j) + r(j) * (u(j)-l(j)); %x(j)是在[l(j),u(j)]均匀分布
    endfor
    if f(x) < min_val:
        min_val = f(x);
        min_x = x;
    endif
endfor
```

- 有约束只需在`if f(x) < min_val`加上判断是否满足约束条件即可

无约束优化方法——间接法

黄金分割法(0.618法)

☞ 一种**一维搜索方法**，单变量函数最优化

☞ **单峰函数**区间上求极小值的一种方法

☞ 基本思想

在搜索区间[a,b]内适当插入两点，将[a,b]分成三段，通过比较这两点的函数值，利用**单峰函数的性质**，就可以**删去最左端或最右端的一段区间。**

然后在留下的区间内再插入一点，重复👆过程。这样可以把包含极小值的区间无限缩小

最速下降法(梯度下降法)

☞ 算法

1. 选定初始点$x^{(0)}$，$k=0.$
2. $k=k+1.$
3. 寻找一个合适的方向$P^{(k)}.$
4. 求出沿$P^{(k)}$方向前进的步长$\lambda^{(k)}$
5. 得到新的点$x^{(k+1)}$，$x^{(k+1)}=x^{(k)}+\lambda^{(k)}P^{(k)}$

检验$x^{(k+1)}$是否最优，如果是最优，则迭代结束，否则转到（2）执行.

👆方向就是**负梯度方向**

$$P^{(k)}=-\nabla f(x^{(k)})$$

如何算最佳步长,$\lambda^{(k)}$,

$$min_{\lambda}f(x^{(k)}+\lambda P^{(k)})\Rightarrow min_{\lambda}f(x^{(k)}-\lambda\nabla f(x^{(k)})$$

算最佳步长属于求关于,λ,的一元函数极值的问题，这一过程被称为一维搜索

最后梯度的长度小于某个阈值，程序停止

☞ 评价

从局部看，梯度下降是朝目标函数值下降最快的方向。但是从全局看，它的收敛是比较慢的

牛顿法

☞ 基本思想

以目标函数的一个**二次函数**作为近似，求这个二次函数的极小值点，以该点来近似原目标函数的一个局部最小值点

☞ 算法

- 给定初始点,$x^{(1)}\in R^n$,给定误差,$\varepsilon>0$,,令,$k=1$,
- 由,$\nabla^2 f(x^{(k)})d^{(k)}=-\nabla f(x^{(k)})$,，解线性方程组得到搜索方向,$d^{(k)}$,
- 令,$x^{(k+1)}=x^{(k)}+d^{(k)}$,
- 如果,$||d^{(k)}||<\varepsilon$,，则迭代终止，否则，置,$k=k+1$,，继续搜索迭代

Tips：, $\nabla^2 f(x^{(k)})$,是求Hesse矩阵

👉 评价

- 当目标函数的的梯度和Hesse矩阵易求，且能对初始点给出较好估计时，可以使用牛顿法，需要注意的是**牛顿方向不一定是下降方向**
- 当初始点远离极小值点时，牛顿法可能不收敛

阻尼牛顿法

👉 与牛顿法的区别：增加了沿牛顿方向进行的一维搜索，迭代公式变为

$$x^{(k+1)} = x^{(k)} + \lambda^k d^{(k)}$$

👉 算法

- 给定初始点, $x^{(1)} \in R^n$, ,给定误差, $\varepsilon > 0,$,令, $k = 1,$
- 由, $\nabla^2 f(x^{(k)}) d^{(k)} = -\nabla f(x^{(k)}),$ ，解线性方程组得到搜索方向, $d^{(k)},$
- 从, $x^{(k)}$,出发，沿方向, $d^{(k)}$,作一维搜索求得最佳搜索步长, $\lambda^{(k)},$

$$min_\lambda f(x^{(k)} + \lambda d^{(k)})$$

- 令, $x^{(k+1)} = x^{(k)} + \lambda^{(k)} d^{(k)},$
- 如果, $||d^{(k)}|| < \varepsilon,$ ，则迭代终止，否则，置, $k = k + 1,$ ，继续搜索迭代

有约束优化方法

👉 主要思想：转换为无约束优化问题进行求解

外点罚函数法

👉 基本思想：利用目标函数和约束条件组成**辅助函数**，对违反约束的点(即位于可行域之外)在辅助函数中加入相应的"惩罚"，使得辅助函数取值很大，而对可行域内的点不予惩罚，此时辅助函数等于原问题的目标函数

👉 对于**只含等式约束的优化问题**

$$minf(x) \ s.t. h_j(x) = 0(j = 1, 2, \cdots, l)$$

建立如下的罚函数

$$F(x, m_k) = f(x) + m_k \sum_{j=1}^{l} h_j^2(x) \ m_k\text{是第}k\text{步迭代时很大的正整数}$$

这样就将有约束转化为了无约束的问题

$$minF(x, m_k)$$

👆的最优解必然使得, $h_j(x) \approx 0,$

👉 对于**只含不等式约束的优化问题**

$$minf(x) \ s.t. g_i(x) \geq 0(i = 1, 2, \cdots, m)$$

构造罚函数

$$F(x, m_k) = f(x) + m_k \sum_{i=1}^{m} g_i^2(x) u_i(g_i) \ m_k\text{是第}k\text{步迭代时很大的正整数} \ u_i(g_i)\text{是单位阶跃函数} \ u_i(g_i) = \{ 0, \quad g_i \geq 0 \ 1, \quad g_i < 0$$

👉 对于**既含等式约束 又含不等式约束的优化问题**

构造罚函数

$$F(x, m_k) = f(x) + m_k P(x) \ P(x) = \sum_{j=1}^{l} L(h_j(x)) + \sum_{i=1}^{m} U(g_i(x)) \text{ 当}y = 0\text{时，}L(y) = 0 \text{ 当}y \neq 0\text{时，}L(y) > 0 \text{ 当}y \geq 0\text{时，}U(y) = 0 \text{ 当}y < 0\text{时，}U(y) > 0$$

关于函数, L和U,的典型取法

$$L(h_j(x)) = |h_j(x)|^\beta \quad \beta \geq 1 \ U(g_i(x)) = [max0, -g_i(x)]^\alpha \quad \alpha \geq 1$$

其中常取, $\alpha = 2, \beta = 2,$

👉 求解步骤

① 初始化

- 选定初始点, $x^{(0)},$
- 初始罚因子, $m_1,$
- 设置罚因子放大系数, $c > 1,$
- 置, $k = 1,$

② 迭代

以,$x^{(k-1)}$,为初始点，求解无约束问题，设其极小值点为,$x^{(k)}$,

- 若,$m_kP(x)<\varepsilon$,，则,$x^{(k)}$,是最优解，迭代终止，否则置,$m_{k+1}=cm_k,k=k+1$,

👉 综上 ,$m_kP(x)$,称为罚项，,m_k,称为罚因子，,$F(x,m_k)$,称为罚函数

👉 转化为无约束问题之后，就可以用无约束的求解方法进行处理。像这种通过求解无约束问题来获得约束问题最优解的方法称为**序列无约束极小化方法(SUMT)**

👉 若**可行域外性质复杂或无定义**，则外点罚函数法失效

内点罚函数法

👉 内点法要求迭代过程始终在可行域内进行，因此将初始点取在可行域内，并若迭代点靠近边界点时，给出的新的目标函数值迅速增大，从而使迭代点留在可行域内

👉 由于内点罚函数总是从内点出发，并保持在可行域内进行搜索，因此内点法适用于**没有等式约束的优化问题**

👉 类似于外点罚函数设计辅助函数的思路，定义**障碍函数**

$$F(x,r_k)=f(x)+r_kB(x)\ B(x)\text{使连续函数，当}x\text{趋向可行域边界时，}B(x)\to+\infty\ r_k\text{是第}k\text{次迭代时的障碍因子，}r_k\text{是很小的整数}$$

$B(x)$,的一般取法

$$B(x)=\sum_{i=1}^{m}\frac{1}{g_i(x)}\ B(x)=-\sum_{i=1}^{m}\ln g_i(x)\ B(x)=\sum_{i=1}^{m}\frac{1}{g_i^2(x)}$$

从而可以将有约束转化为无约束

👉 求解步骤和外点罚函数法类似，但是需要设置的是**障碍因子的缩小系数**

👉 内点法和外点法的收敛情况与**罚因子的选取关系很大**

动态规划

👉 是用来解决**多阶段决策过程最优化的一种数量方法。**可以把一个,n,维决策问题变换为几个一维最优化问题

👉 例子

- 最短路
- 投资分配
- 背包问题(也可归为整数规划)
- 排序问题

👉 用LINGO求解最短路

目标规划和多目标规划

多目标规划

👉 标准模型

$$\begin{array}{lll}
min/max & f_m(x), & m=1,2,\dots,M \\
subject\ to & g_j(x)\ge 0, & j=1,2,\dots,J \\
 & h_k(x)=0, & k=1,2,\dots,K \\
 & \underset{\substack{lower\\bound}}{x_i^{(L)}}\le x_i\le \underset{\substack{upper\\bound}}{x_i^{(U)}}, & i=1,2,\dots,n
\end{array}$$

👉 特点

- 包含多个**有冲突**的目标函数
- 希望能找到很好平衡全部优化目标的解集

帕累托最优(Pareto Optimal)

👉 是指**资源分配的一种理想状态。**

- 给定固有的一群人和可分配的资源，如果从一种分配状态到另一种状态的变化中，**在没有使任何人情况变坏的前提下**，使得至少一个人变得更好，称为**帕累托改善**
- 帕累托的**最优状态**即为不可能再有更多的帕累托改善的状态

👉 一些概念

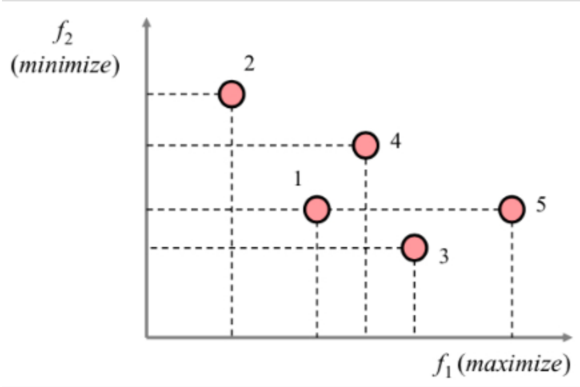
- 支配(Dominace)

  当,x_1,x_2,满足下述条件时，称,x_1,支配,x_2,

  - 对于所有的目标函数,,x_1,不比,x_2,差

  - 至少在一个目标函数上，,x_1,严格比,x_2,好

如下图中，点1支配点2，点5支配点1，点1和点4互不支配



- 不可支配解集(Non-dominated solution set)

  当一个解集中**任何一个解**都不能被该集合中其他解**支配**，则称该解集为**不可支配解集**

- 帕累托最优解集(Pareto-optimal set)

  所有可行的不可支配解集

- 帕累托最优前沿面(Pareto-optimal front)

  帕累托最优解集的边界

👉 MOO问题的目标

- 寻找尽可能接近帕累托最优前沿面的解集
  - 因为不一定可以找到不可支配解集，定义了**帕累托等级**
  - 尽可能增大找到解的多样性

👉 帕累托等级

在一组解中，**不可支配解的帕累托等级为1**，将不可支配解从解的集合中**删除**，类推等级为2,3,...解

MOO的传统解法

- 加权法

  - 很难设定一个权重向量

$$
\begin{aligned}
min \quad & F(x) = \sum_{m=1}^{M} w_m f_m(x), \\
subject\ to \quad & g_j(x) \geq 0, \quad j = 1, 2, \ldots, J \\
& h_k(x) = 0, \quad k = 1, 2, \ldots, K \\
& x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \ldots, n
\end{aligned}
$$

  - 在一些非凸情况不能保证获得帕累托最优解

- $,\varepsilon - constraint, method,$

  - 只保留一个目标函数，其他的目标函数被设定为值约束

$$
\begin{aligned}
min \quad & f_\mu(x), \\
subject\ to \quad & f_m(x) \leq \epsilon_m, \quad m = 1, 2, \ldots, M\ and\ m \neq \mu \\
& g_j(x) \geq 0, \quad j = 1, 2, \ldots, J \\
& h_k(x) = 0, \quad k = 1, 2, \ldots, K \\
& x_i^{(L)} \leq x_i \leq x_i^{(U)}, \quad i = 1, 2, \ldots, n
\end{aligned}
$$

  - 保留哪一个函数要精心选择

多目标遗传算法

👉 NSGA2(Nondominated sorting genetic algorithm 非支配排序遗传算法)

👉 利用遗传算法来搜索帕累托最优前沿面

👉 算法思想

- 随机产生一定规模的种群，**不可支配排序**后通过遗传算法的选择、交叉、变异三个基本操作得到第一代**子代种群**
- 从第二代开始，将父代种群与子代种群合并，进行快速非支配排序，同时对每个不可支配层中的个体进行拥挤度计算，根据**不可支配关系**和**个体的拥挤度**选取合适的个体组成新的**父代种群**，继续进行遗传算法的三步操作，直到满足程序结束的条件

👉 算法流程

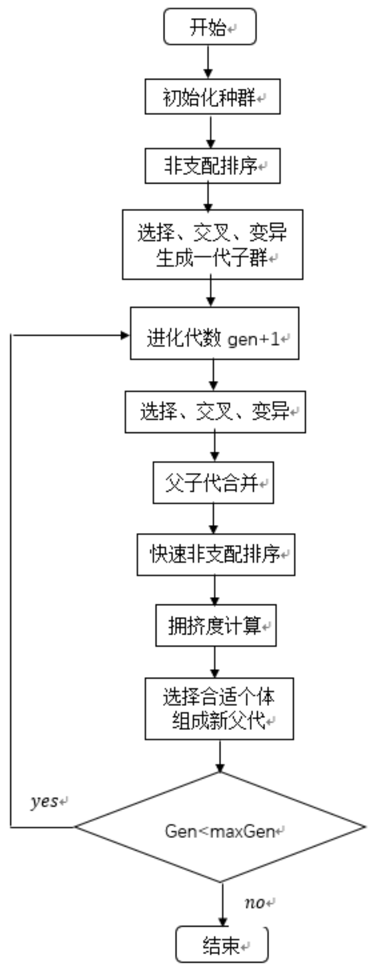


图 3 NSGA-Ⅱ算法流程图

快速不可支配排序

👉 需要先计算每个个体,p,的**被支配个数**,n_p,和该个体**支配的解**的集合,S_p,,伪代码如下

计算n_p, S_p将种群中$n_p=0$的个体放入集合F_1中 $for, i, in, F_1:$ for, j, in, S_i : 注意是找被个体i支配的解j $n_j = n_j - 1$即消除等级1的解 对其余个体的支配 相当于删

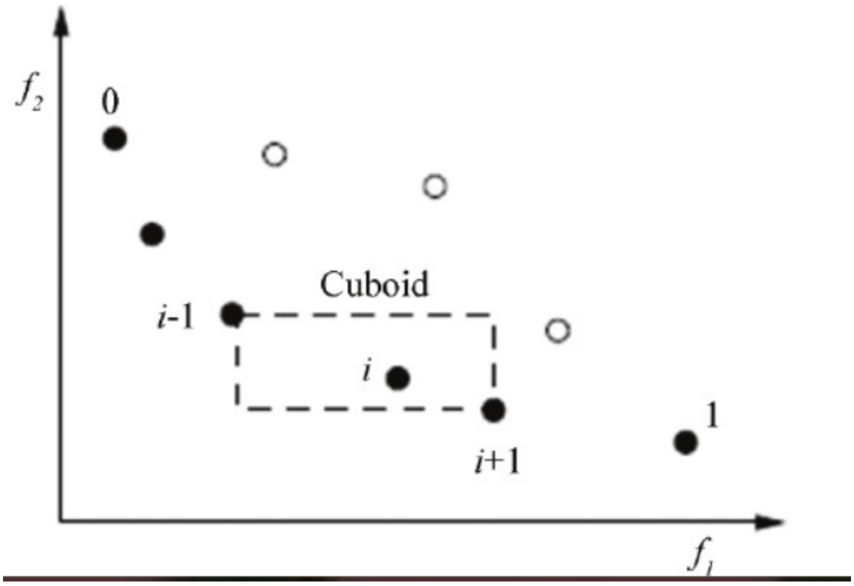
拥挤度

👉 同一等级的非支配个体集合中，为了保证解的个体能**均匀分布在帕累托前沿**，即要使同一层中的非支配个体具有**多样性**，因此引入**拥挤度(拥挤距离)**，用于计算同一等级的集合中某个给定个体周围其他个体的密度

👉 显然拥挤度越高，个体的多样化程度也越高

👉 每个个体拥挤距离的是计算**与其相邻的两个个体**在**每个子目标函数**上的距离差之和来求取，**注意，每个子目标函数做差之后要进行归一化处理，即除以次子目标函数的极差**

👉 举例来说，对于二目标规划，拥挤距离就是下图中矩形的长宽之和



👉 算法流程

$$设同一等级的非支配个体数目为N，每个个体的拥挤度为n_d(n \in 1,2,\cdots,N)$$

$n_d = 0 for,,$ 每个子目标函数 f_m ： 根据子目标函数对此等级的个体排序，记 f_m^{max} 为此等级个体在子目标函数 f_m 上的最大值，f_m^{min} …… ∞ $n_d = n_d + \frac{f_m}{}$

选择合适的父代——精英策略

将父代种群 C_i 和子代种群 D_i 合成种群 R_i 生成新的父代种群 C_{i+1} ①根据帕累托等级从低到高的顺序，将整层种群放入 C_{i+1} …… 父代种群 C_{i+1} ，直到某一层个体不能全部放入父代种

选择 交叉 变异

👉 采用ga的算法

遗传算法GA

GA基本概念

👉 思想

- 种群中个体的选择
- 种群中交叉繁殖
- 种群中个体的变异

👆反复执行，个体逐渐优化

GA Example

👉 Ex

$$求解方程\quad x^{10} + e^x = 100 (x > 0)$$

将方程求解问题转化为生存问题——将区间[0,10]划分成若干小区间，设想每个小区间为一个生物个体，$,|x^{10} + e^x - 100|,$ 最小的个体有最好的生存能力,该个体即为解

- 初始化

👉 个体编码

若使用二进制编码，假设要求的精度是小数点后m位，区间的长度是n，则至少要把区间 $,n \times 10^m,$ 份，之后求与此数最接近的2的 $,p,$ 次幂， $,p,$ 就是二进制串的位数

假定本题要求小数点后两位，至少要把区间划分 $,10 \times 10^2 = 1000,$ 份，所以要把区间划分为 $,2^{10} = 1024,$ ，二进制串的位数为10

👉 种群初始化

随机生成任意数量的10位二进制串

- 选择

👉 定义适应度函数

这里是以目标函数的倒数作为适应度函数

$$f = \frac{1}{|x^{10} + e^x - 100|}$$

👉 选择适应度较大的个体

轮盘赌：将所有个体的适应度求和，得到总适应度。每一个体被选择到的概率为**该个体适应度/总适应度**

代码实现——产生[0,1]的随机数选择个体

- 交叉

选中的优势个体进行交叉，这里采用**单点交叉：在编码串中只随机设置一个交叉点，以该点为分界，交换部分染色体**

👉 交叉算子有好几种

- 变异

在个体中的某几位的值发生改变

👉 变异的方法也有好几种

👉 交叉和变异操作都以一定概率进行

👉 反复进行选择，交叉，变异并记录最优个体。

👉 程序结束的判定

- 根据循环次数
- 根据最大适应度
- 根据种群中相同个体数与总个体数的比值

GA Solve TSP

- 初始化

取长度为N的数字串，串中数字互不重复，取值范围为[1,N]之间的整数，每一个数字串代表一个个体，个体中数字出现的位置是路径中城市出现的顺序

- 选择

适应度函数——路径长度的导数

- 交叉

TSP的交叉算子与普通的不同——要保证生成的解是有效解

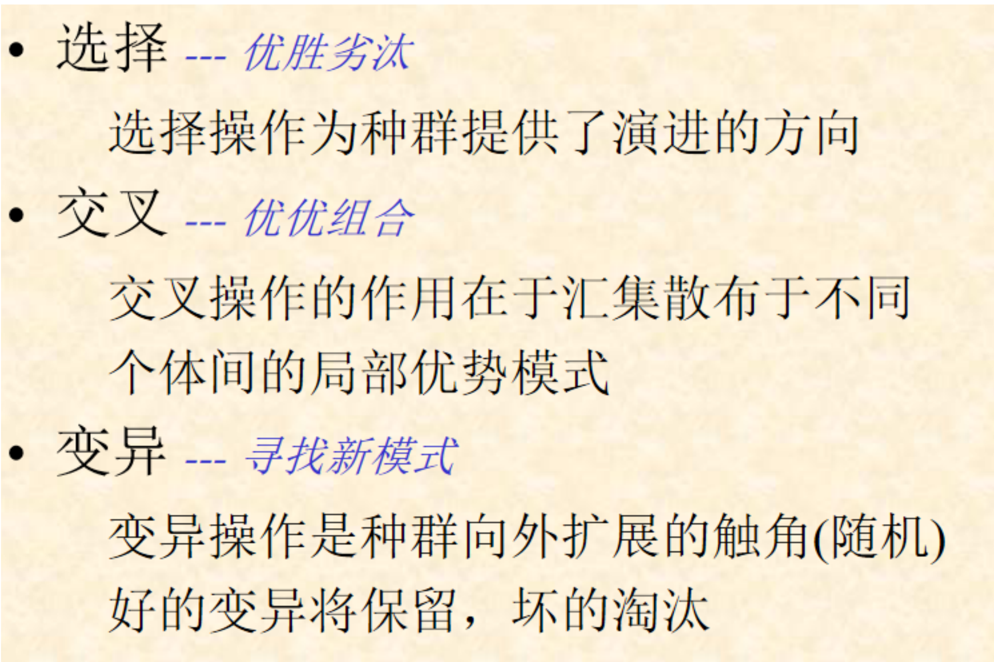
👉 在一个父个体中随机选取一段子串A，在另一个父个体中将字串A出现的数字去掉形成串B，AB就是一个新的字串

- 变异

👉 常用的变异操作——随机选取两个相邻位置的数字，交换顺序

评价GA

👉 GA各步骤的作用



👉 GA的评价

优点

- 全局收敛性(依概率收敛)

- 泛化能力很强

缺点

- 得到的解是**近似的数值解**
- 对于经典数学可以解决的问题，效率很低

模拟退火

👉 不会陷入局部最优，因为会有一定概率不会接收当前最优解(**Metropolis准则**)

$$\text{当前能量为}E(n),\text{产生扰动后在}n+1\text{状态的能量为}E(n+1)\ P(\text{接受新的状态}) = \left\{ 1, \quad E(n+1) <= E(n)\ e^{-\frac{\Delta E}{T}}, \quad E(n+1) > E(n) \quad T\text{是当前状态的温度} \right.$$

类似于RL中的**epsilon greedy**

👉 算法的步骤

```
temperature = 1000; %初始温度
thres_temperature = 1; %设置温度阈值
cooling_rate = 0.94; %温度的下降率
iteration_count = 1; %计数迭代次数
prev_energy = ?; %获得初始解

while tempratrue > thres_temperature
    curr_energy = ?; %扰动获得新的解
    diff = curr_energy - prev_energy;
    %利用metropolis准则
    if (diff<0) || (rand < exp(-diff/temperature))
        prev_energy = curr_energy; %获得新的状态
        iteration_count = iteration_count + 1;
    end
    if iteration_count >= 100 %如果某个T的迭代次数超过100次 降温
        temperature = cooling_rate * temperature;
        iteration_count = 0; %迭代次数置零
```

禁忌搜索(要补充)

Matlab常用的优化函数和优化工具箱(未写)

随机模拟