西安邮电大学

快速精英多目标遗传算法—NSGA-II

刘田，崔栖瑞

目录Contents

01

背景概念

1-1 背景

1-2 Pareto最优解相关概念

01 背景

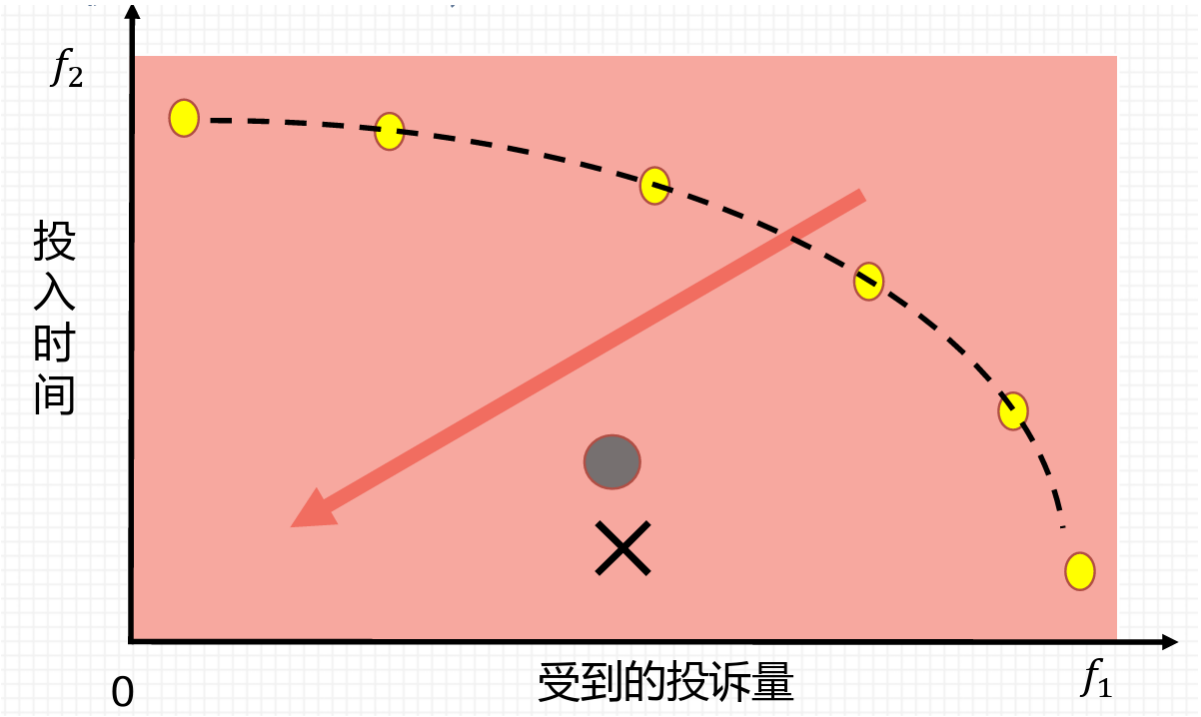
在实际问题中，许多优化问题存在多个相互矛盾的目标，这些目标通常无法简单地归并成一个单一的目标函数。例如，在工程设计中，优化方案可能需要同时考虑成本、性能、可靠性等多个方面；在项目管理中，需要平衡时间、成本和质量等多个指标。在这样的情况下，传统的单目标优化方法往往无法很好地找到满足多个目标的最优解，因为不同目标之间存在冲突。

多目标优化是涉及多个目标函数同时优化的数学问题.

需要在两个或多个相互冲突的目标之间进行权衡的情况下作出最优决策

01 概念

这就有一条权衡曲线，不同的人做不同的决策，最后的目标是f2要小，f1也要小，但无法解决，就产生了决策边界也成为帕累托前沿，对应的边界是帕累托最优边界，解就是帕累托最优解（解并没有谁好谁坏，要根据不同情况做不同的决策）



02

算法原理

2-1 基本原理

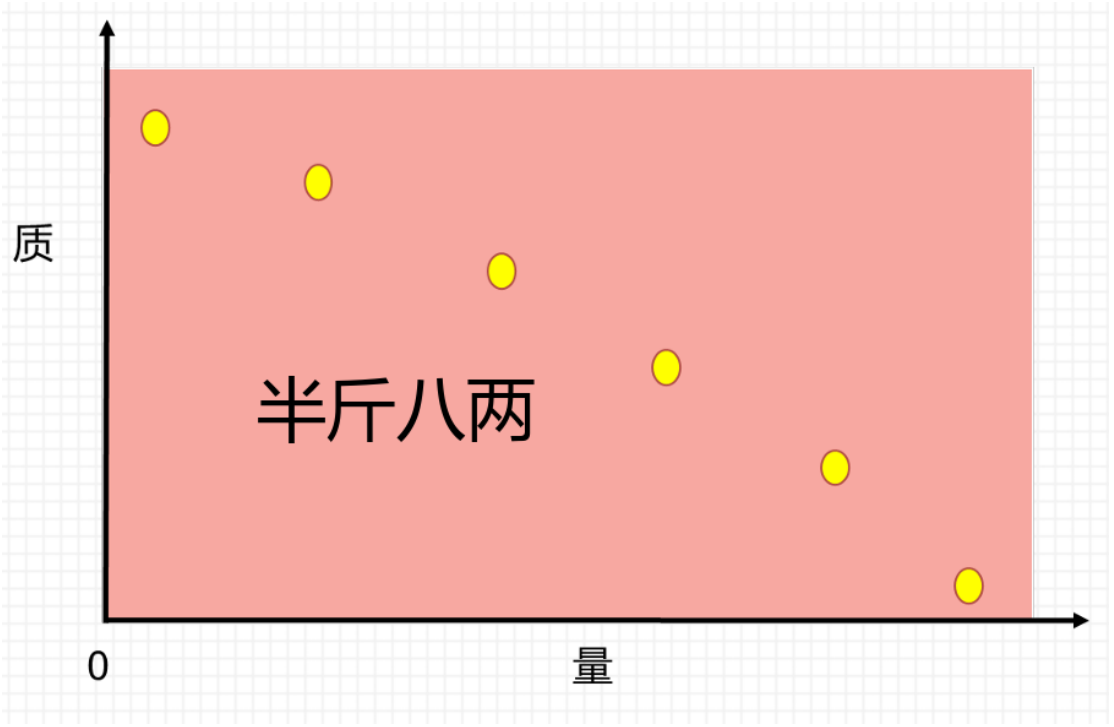
2-2 非支配排序

2-3 拥挤距离

01 基本原理

智能优化基本流程

- 初始化N个解(个体)
- 计算函数值(或者适应度)
- 利用旧解产生新解
  - 各种策略：GA，PSO，灰狼优化等(本质一样)
- 选择得到新一轮的解
  - 单目标：谁的y值小，谁就好(最小化问题)
    - 旧的解50个，新的解50个，直接整体排序，保留50个即可
  - 多目标：
    - 旧的解50个,新的解50，怎么排序？谁更好？



多目标优化真正的核心难点：怎么评价解得好坏，即怎么排序

02 非支配排序

	X	Y
Max Obj1	4	4
Max Obj2	0.3	0.2

解x是否优于解y

是

	X	Y
Max Obj1	5	4
Max Obj2	0.1	0.25

解x是否优于解y

否

可以看到上面左一表格中Max obj1中x和y的值相同，但在Max obj2中x的值大于y的值，所以说x支配于y

但另一个表格其X和Y的值在Max obj1与Max obj2互有大小关系，所以并不能说x支配y

02 非支配排序

B≺D、A≺D(以右图$minimize$为例)

A与B，互相无法支配(看成一样的)

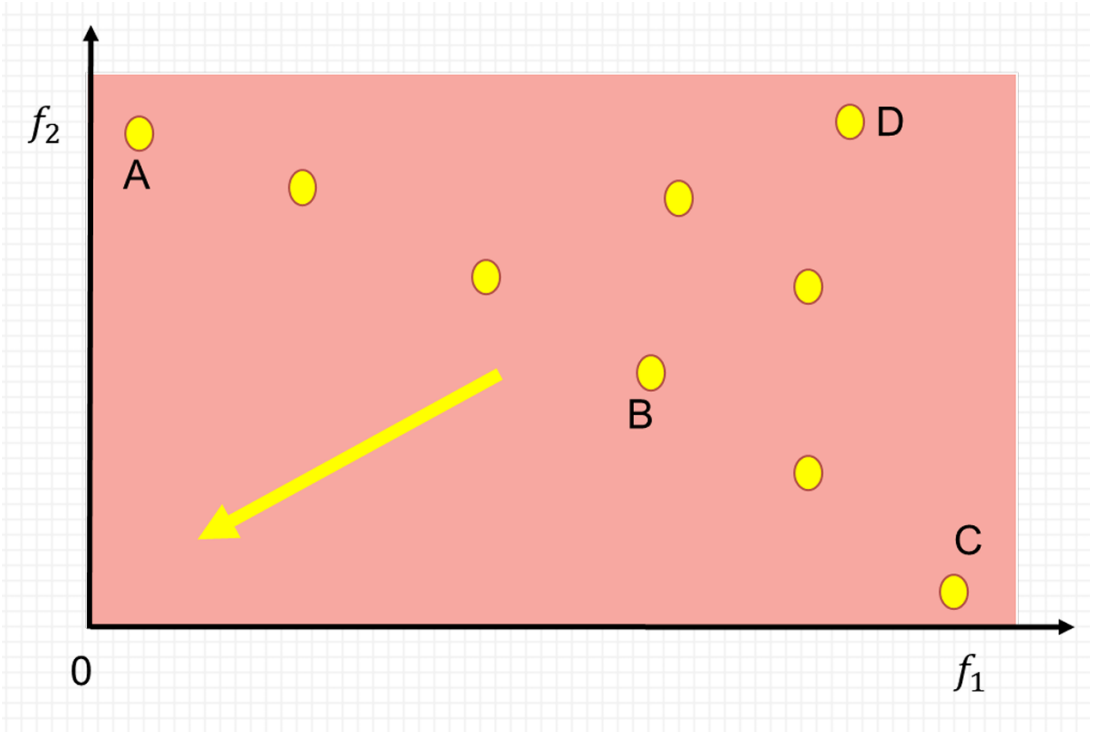
代码实现: flag = all(A<=B) && any(A<B)

代码代表意思：所有a都小于等于b，并且至少一个a小于b

A支配于D，想说的是A严格比D好

通过点之前的支配关系就能知道其排序的顺序

但非支配排序只解决了一般的排序，但同一个帕累托前言上的a和b怎么判断?



02 拥挤距离

几何角度理解拥挤距离：

以两个目标函数为例，右图中黑点和白点分别为两个非支配前沿，对于解i，从与i在同一非支配前沿中选择与解i最相近的两个点i-1和i+1为顶点组成一个长方形，拥挤距离即为长方形平均边长

算的是两个解左右两边绝对值的距离

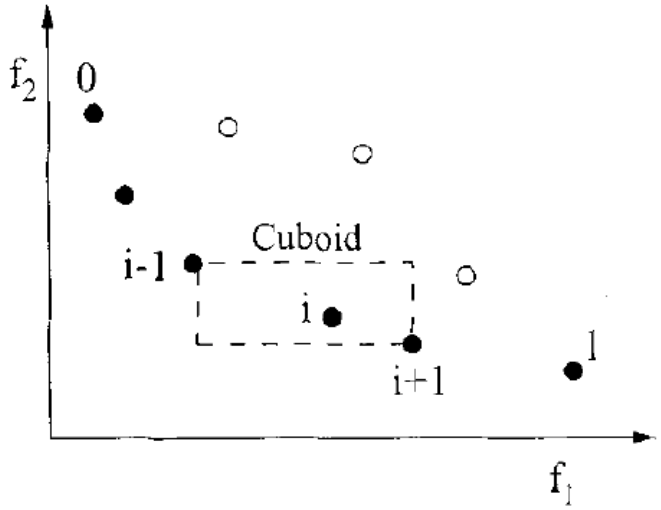


Fig. 1. Crowding-distance calculation. Points marked in filled circles are solutions of the same nondominated front.

02 拥挤距离

拥挤距离越大:

这里解比较空旷

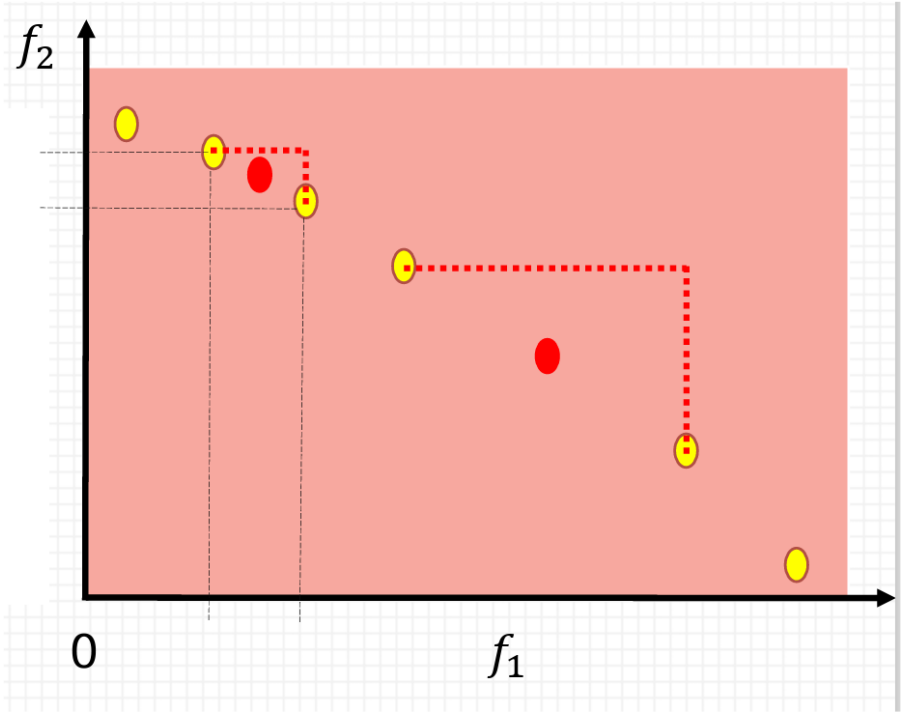
选择拥挤距离大的解，有利于种群多样性

因此，非支配解之间，也能进行比较大小

去掉拥挤距离小的解，让其均匀撒在帕累托前沿上

比较大小的问题，得到了解决

拥挤距离是描述基因个体与其相邻个体之间拥挤度的指标，拥挤距离越大表明种群中个体的分布越分散



03

算法分析

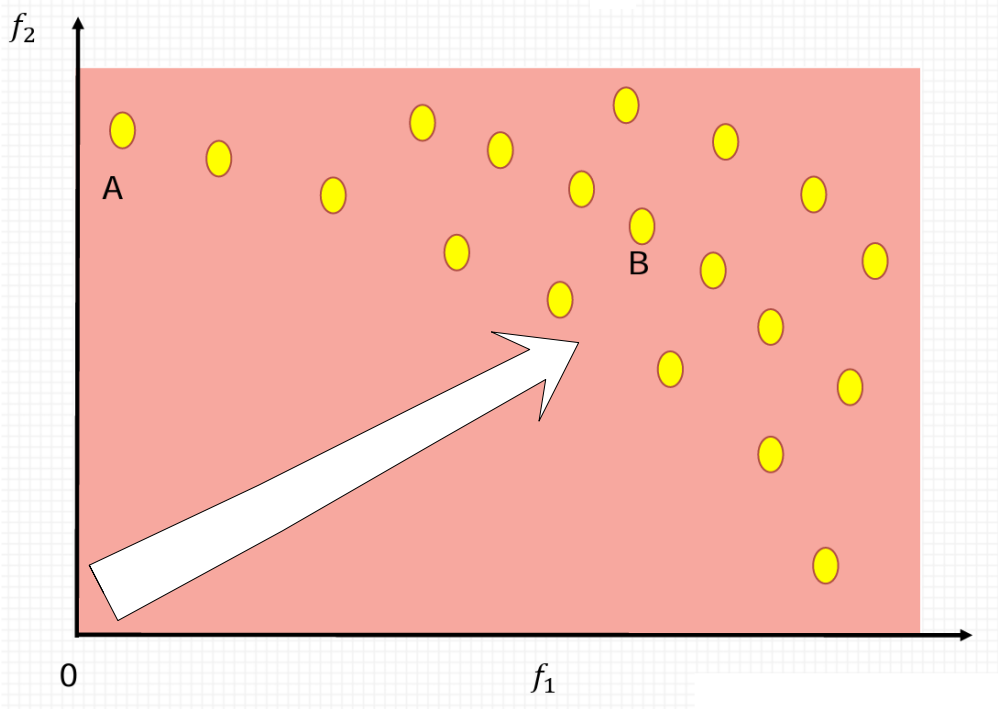
3-1 非支配排序

3-2 拥挤距离

03 非支配排序

NSGA-I的排序

```
Pop = pop
While Pop :
    for pi in Pop :
        for pj inPop:
            if pj ≺pi : pi.dominatecount += 1
    S = find
    pi. dominatecount == 0
    S .Rank = i          # 标志这是第几层
    F .append(S)         # 记录每一层的个体
    Pop = Pop - S        # 下一轮两两比较
    i += 1
```



最坏复杂度O(MN3),M是obj个数，N是pop size

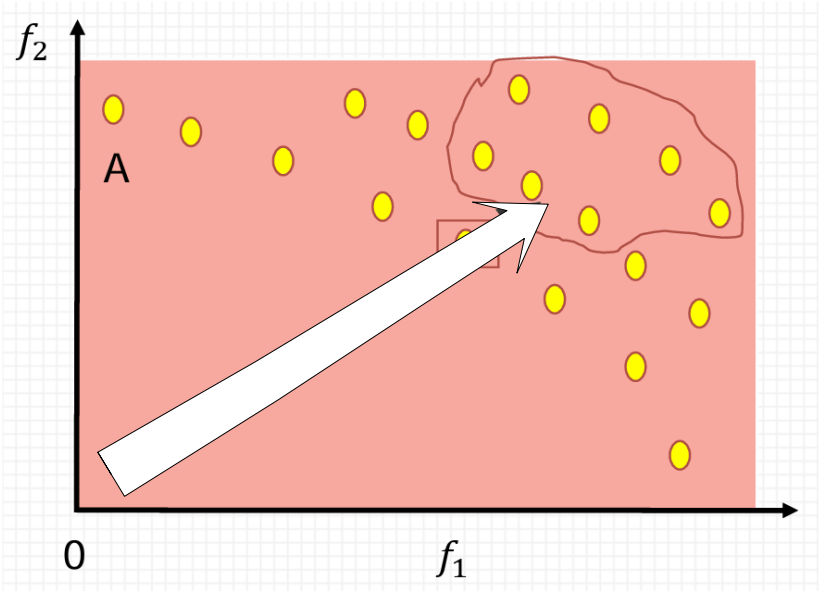
03 非支配排序

NSGA-II的排序

fast non dominated sort(P)

For each p in P :　　# 对种群 P 中的每个解 p
  Set $S_p = \phi$.　　# 初始化被支配集合S_p为空集
  Set $n_p = 0$.　　# 初始化支配数n_p为 0
  For each q in P:　　# 对种群 P 中的每个解 q
    If p dominates q:　　# 如果 p 支配 q
      Then $S_p = S_p \cup \{q\}$　　# 添加 q 到 p 的被支配集合S_p
    Else if q dominates p:　　# 如果 q 支配 p
      Then $n_p = n_p + 1$　　# 增加 p 的支配数n_p+1
  If $n_p = 0$:　　# 如果 p 的支配数n_p为 0
    Then $p_{rank} = 1$　　# 设置 p 等级为 1
    And $F_1 = F_1 \cup \{p\}$　　# 将 p 增加到第一层非支配前沿F_1

Set i = 1　　# i 代表当前非支配前沿索引
While $F_i \neq \emptyset$　　# 当非支配前沿为空时结束
  Set $Q = \phi$　　# 初始化Q用于存储第 i+1 层的非支配解
  For each p in F_i:　　# 对第 i 层非支配前沿F_i中的每个解 p
    For each q in S_p:　　# 对解 p 的被支配集合中的每个成员 q
      Then $n_q = n_q - 1$　　# 减小 q 的支配数$n_q - 1$
      If $n_q = 0$:　　# 如果 q 的支配数为 01
        Then $q_{rank} = i + 1$　　# 设置 q 的等级为 i+1
        And $Q = Q \cup \{q\}$　　# 将 q 添加到集合 Q 中
  Set i = i + 1　　# 开始循环 i+1 层非支配前沿
  Set $F_i = Q$　　# 第 i+1 层非支配前沿为集合 Q



Sp={…}，当前个体p所支配的那些个体

Np=$count$,能够支配当前个体p的个数

最坏复杂度O($MN2$)

M是obj个数，N是pop size

03 拥挤距离

crowding distance assignment ($\mathcal{L}$)

Set $l = |\mathcal{L}|$ # 非支配前沿$\mathcal{L}$的解数量

For each i, set $\mathcal{L}[i]_{distance} = 0$ # 初始化$\mathcal{L}$中每个解的拥挤距离

For each objective m: # 对每个目标函数 m

Sort $\mathcal{L} = sort(\mathcal{L}, m)$ # 按照目标函数值 m 给$\mathcal{L}$中的解排序

目标值最大的点和最小的点为边界解，拥挤距离设为无穷

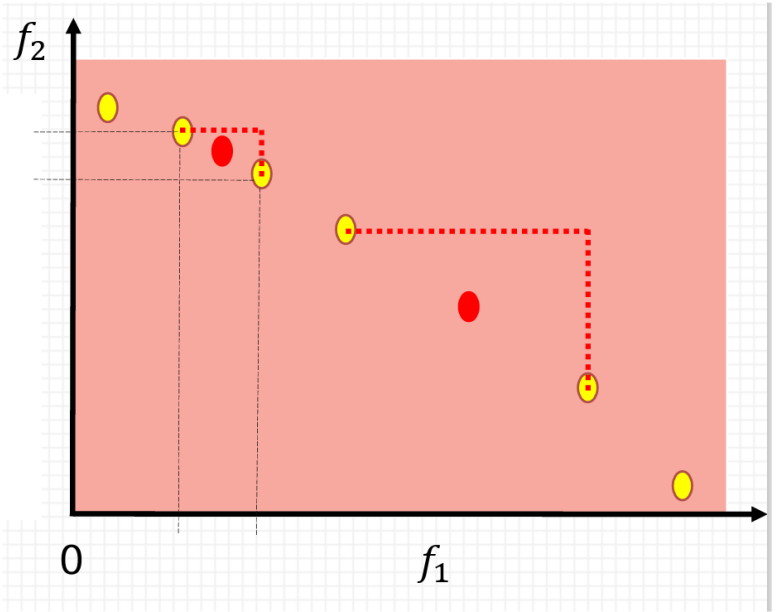
Set $\mathcal{L}[1]_{distance} = \mathcal{L}[l]_{distance} = \infty$

For i = 2 to $l - 1$: # 对$\mathcal{L}$中的每个解，除开边界解

每个目标函数下的距离值为前后点目标值差值归一化结果

完整的拥挤距离为所有目标函数下的距离值之和

$\mathcal{L}[i]_{distance} = \mathcal{L}[i]_{distance} + (\mathcal{L}[i+1].m - \mathcal{L}[i-1].m)/(f_m^{max} - f_m^{min})$



03 拥挤距离

拥挤度比较算子

在计算完每个解的拥挤距离后，从一定程度上来说，某个解的拥挤距离越小，这个解被其他解拥挤的程度越高。接下来通过拥挤度比较算子来选择解实现更广的帕累托最优解分布。种群中的每个个体都有两个属性：

非支配等级rank：1是最高等级

拥挤距离distance

定义一个比较顺序，满足一下判断条件。对于不同非支配等级的两个解，倾向于选择rank值更低的解，如果两个解的等级相同，倾向于选择拥挤距离更大或者说拥挤区域更小的解。

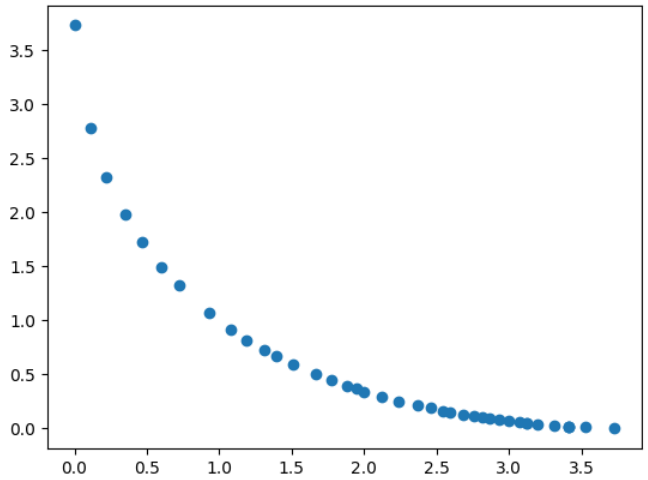
crowded comparison operator (i, j)

$$i \prec_n j \text{ if } (i_{rank} < j_{rank}) \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance}))$$

04

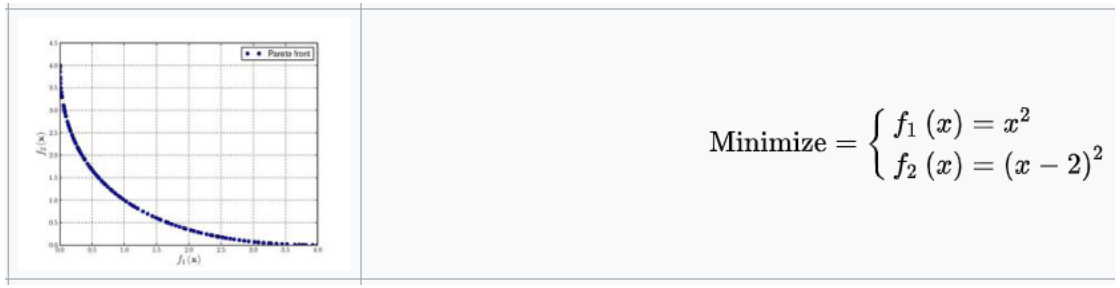
实验运行成果

04 实验结果展示



运行结果

测试函数链接：https://en.wikipedia.org/wiki/Test_functions_for_optimization

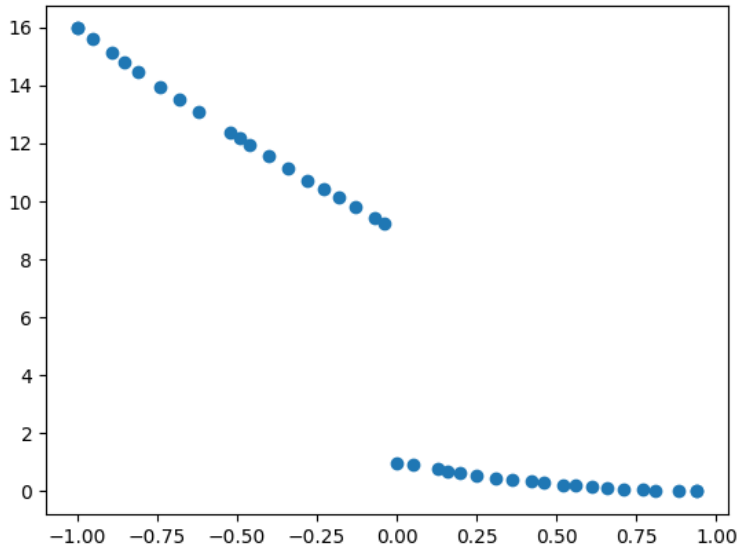


测试函数及结果

```python
# 测试函数
def function1(x):
    y = x ** 2
    return round(y, 2)
def function2(x):
    y = (x - 2) ** 2
    return round(y, 2)
```

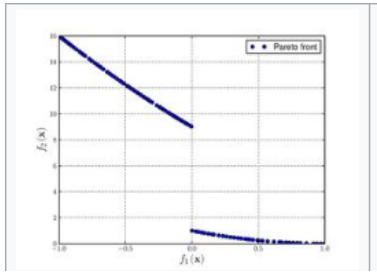
仓库链接：https://github.com/liuslient/Test/tree/master/nsga

04 实验结果展示



运行结果

测试函数链接：https://en.wikipedia.org/wiki/Test_functions_for_optimization



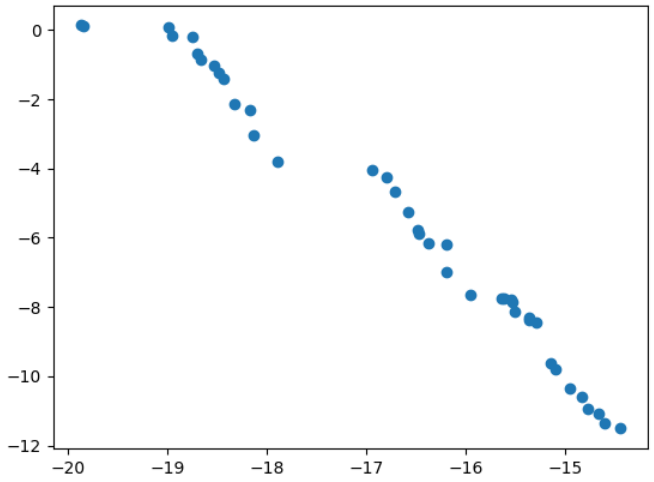
$$\text{Minimize} = \begin{cases} f_1(x) = \begin{cases} -x, & \text{if } x \leq 1 \\ x-2, & \text{if } 1 < x \leq 3 \\ 4-x, & \text{if } 3 < x \leq 4 \\ x-4, & \text{if } x > 4 \end{cases} \\ f_2(x) = (x-5)^2 \end{cases}$$

测试函数及结果

```python
# 测试函数
def function1(x):
    if x <= 1:
        y = -x
    elif x <= 3:
        y = x - 2
    elif x <= 4:
        y = 4 - x
    else:
        y = x - 4
    return round(y, 2)
def function2(x):
    y = (x - 5)**2
    return round(y, 2)
```

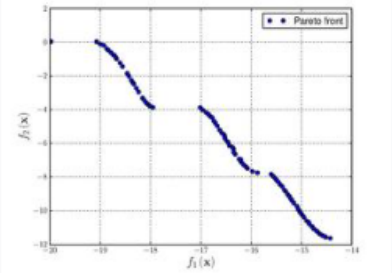
仓库链接：https://github.com/liuslient/Test/tree/master/nsga

04 实验结果展示



运行结果

测试函数链接：https://en.wikipedia.org/wiki/Test_functions_for_optimization



$$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) = \sum_{i=1}^{2} \left[-10 \exp\left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right] \\ f_2(\boldsymbol{x}) = \sum_{i=1}^{3} \left[|x_i|^{0.8} + 5 \sin\left(x_i^3 \right) \right] \end{cases}$$

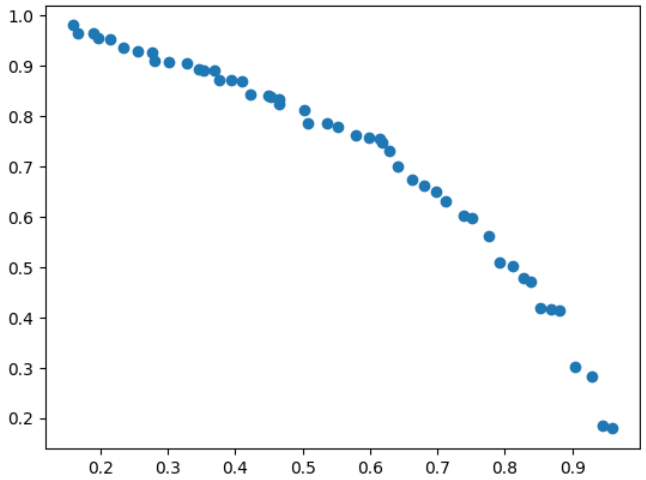
测试函数及结果

```python
# 测试函数
def function1(x):
    return sum(-10 * np.exp(-0.2 * np.sqrt(x[i]**2 + x[i+1]**2)) for i in range(2))

def function2(x):
    return sum(abs(x[i])**0.8 + 5 * np.sin(x[i]**3) for i in range(3))
```

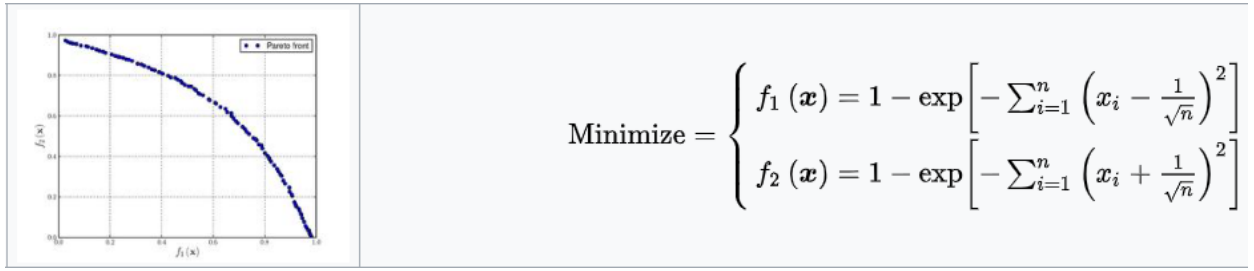
仓库链接：https://github.com/liuslient/Test/tree/master/nsga

04 实验结果展示



运行结果

测试函数链接：https://en.wikipedia.org/wiki/Test_functions_for_optimization



测试函数及结果

```python
# 测试函数
def function1(x):
    n = len(x)
    return 1 - np.exp(-sum(((x[i] - 1) / np.sqrt(n))**2 for i in range(n)))

def function2(x):
    n = len(x)
    return 1 - np.exp(-sum(((x[i] + 1) / np.sqrt(n))**2 for i in range(n)))
```

仓库链接：https://github.com/liuslient/Test/tree/master/nsga

西安邮电大学

请老师批评指正

Thank You