计算智能选讲

武建

wujian@sxufe.edu.cn

内容：

- 搜索最优解的群体策略
- 群体策略的遗传算法实现

■ 搜索最优解的群体策略

- 问题
- 建立目标函数
- 建立群体
- 群体优劣评价
- 迭代搜索最优解

1 问题

$$AX = b$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

求解方程组的近似最优解。

2 建立优化目标函数

$$Obj = \| AX - b \|$$

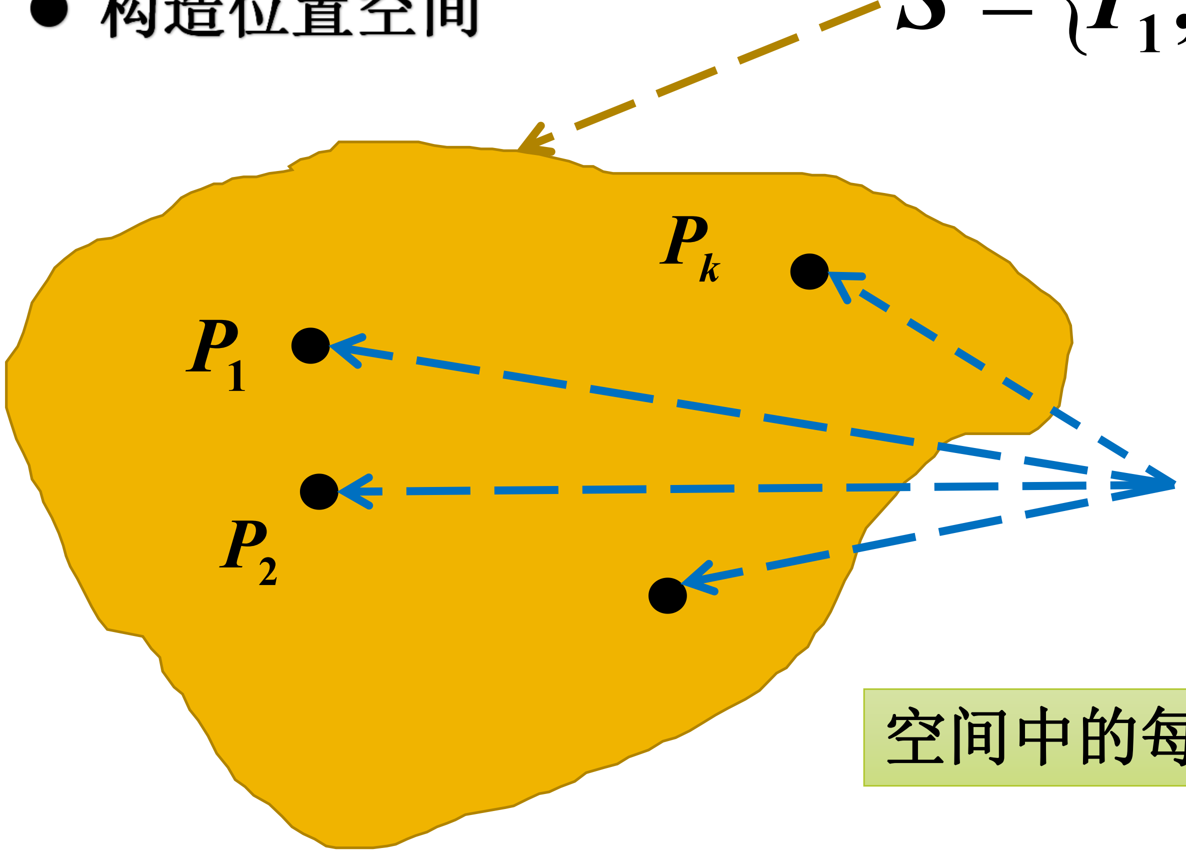
范数

目标函数值越小，搜索得到的解向量越好

3 构造搜索群体

- 构造位置空间

$$S = \{P_1, P_2, P_3, \ldots\ldots\}$$



位置向量

$$P = (p_1, p_2, \ldots, p_n)^T, p_i \in R$$

空间中的每个点对应一个位置向量。

位置向量的维数与解向量的维数相同。

- **构造随机向量**

个体向量或个体

$$X=(x_1,x_2,\cdots,x_n)\xrightarrow{\text{取值}}P=\left(p_1,p_2,\ldots,p_n\right)^T\in S$$

随机变量 $$x_i\xrightarrow{\text{取值}}p_i\in[c_i,d_i],i=1,2,\ldots,n$$

每个个体的一次取值对应一个可能得解向量

- 构造群体向量组或矩阵

$$Population = \{X_1, X_2, ..., X_N\}$$

或

$$Population = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix}$$

群体的一次取值对应多个可能得解向量

4 群体优劣评价

计算每个个体的目标函数值，称作适应度：

$$Obj = \| AX - b \|$$

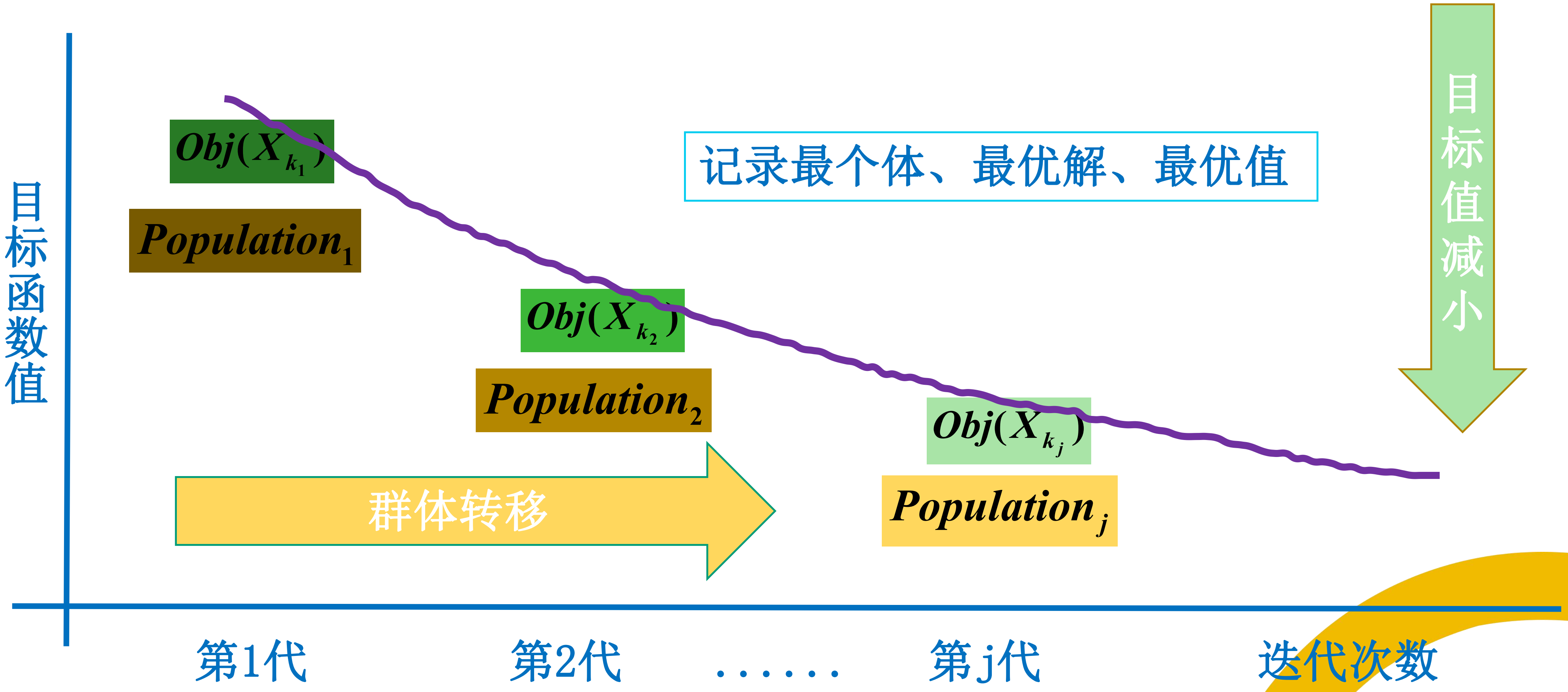
$$Population = \{X_1, X_2, \dots, X_N\}$$

$$F(X_i) = \frac{1}{Obj(X_i) + \varepsilon}, i = 1,2,\ldots,N$$

根据适应度确定最优个体，及其取值：

$$X = (x_1, x_2, \cdots, x_n) \qquad P = (p_1, p_2, \ldots, p_n)^T \in S$$

5 迭代搜索最优解



目标函数值
$Obj(X_{k_1})$
$Population_1$
$Obj(X_{k_2})$
$Population_2$
$Obj(X_{k_j})$
$Population_j$
记录最个体、最优解、最优值
目标值减小
群体转移
第1代
第2代
……
第j代
迭代次数

■ 遗传算法

- 建立目标函数
- 建立群体
- 群体转移
  - ◆ 遗传算子

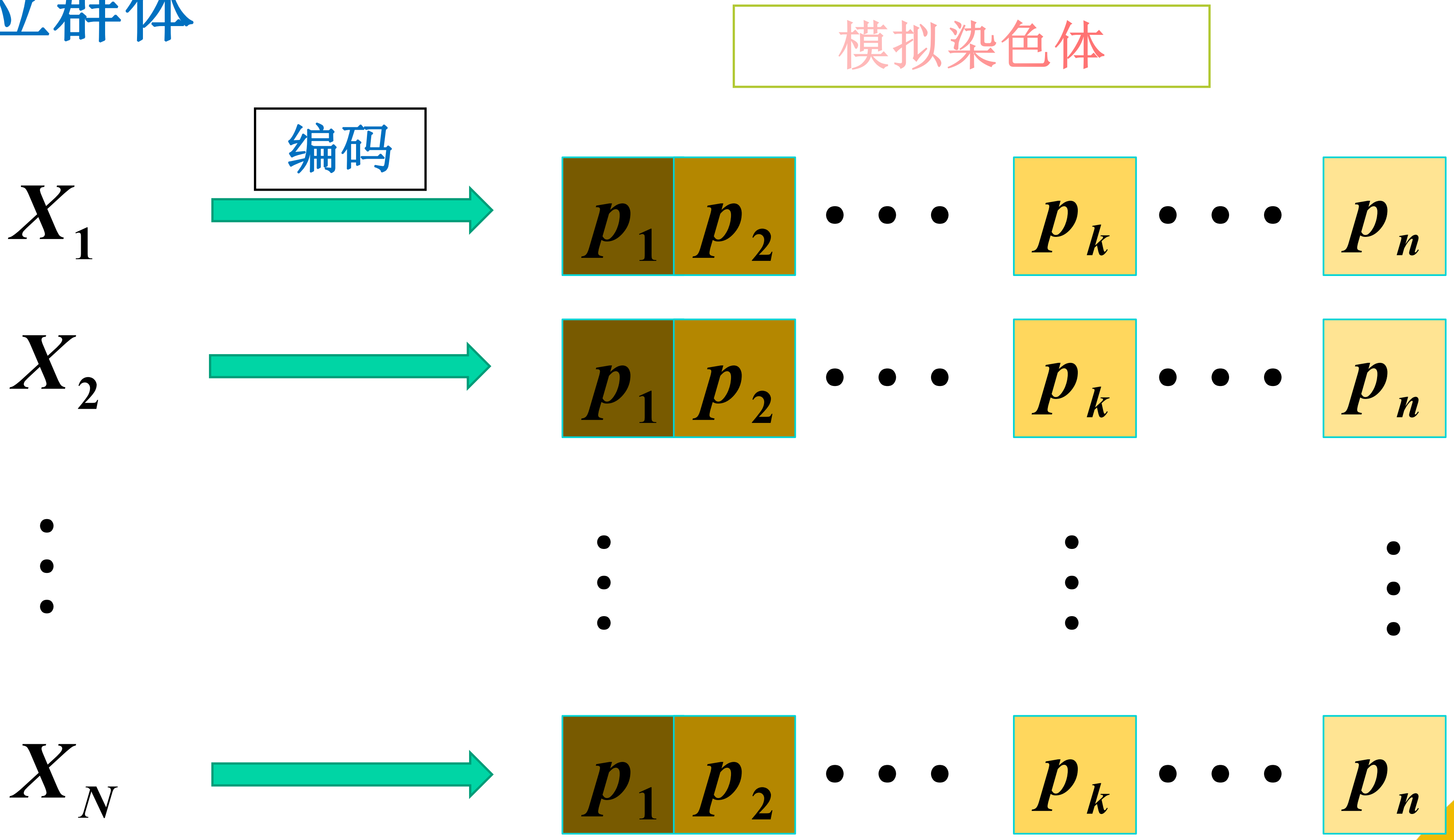
1 建立优化目标函数

$$Obj = \| AX - b \|$$

范数

目标函数值越小，搜索得到的解向量越好

2 建立群体



3 群体转移：选择算子

$p_1\ p_2 \cdots p_k \cdots p_n$

解码

$$F_1 = \frac{1}{Obj((p_1, p_2, \dots, p_n))}$$

$p_1\ p_2 \cdots p_k \cdots p_n$

适应度值

$\vdots \qquad \vdots \qquad \vdots$

解码

$$F_i = \frac{1}{Obj((p_1, p_2, \dots, p_n))}$$

$p_1\ p_2 \cdots p_k \cdots p_n$

3 群体转移：选择算子

✓从父代群体中选取一些个体，遗传到下一代群体

✓轮盘赌选择法：　个体被选中的概率与其适应度值大小成正比

F_i

✓步骤：

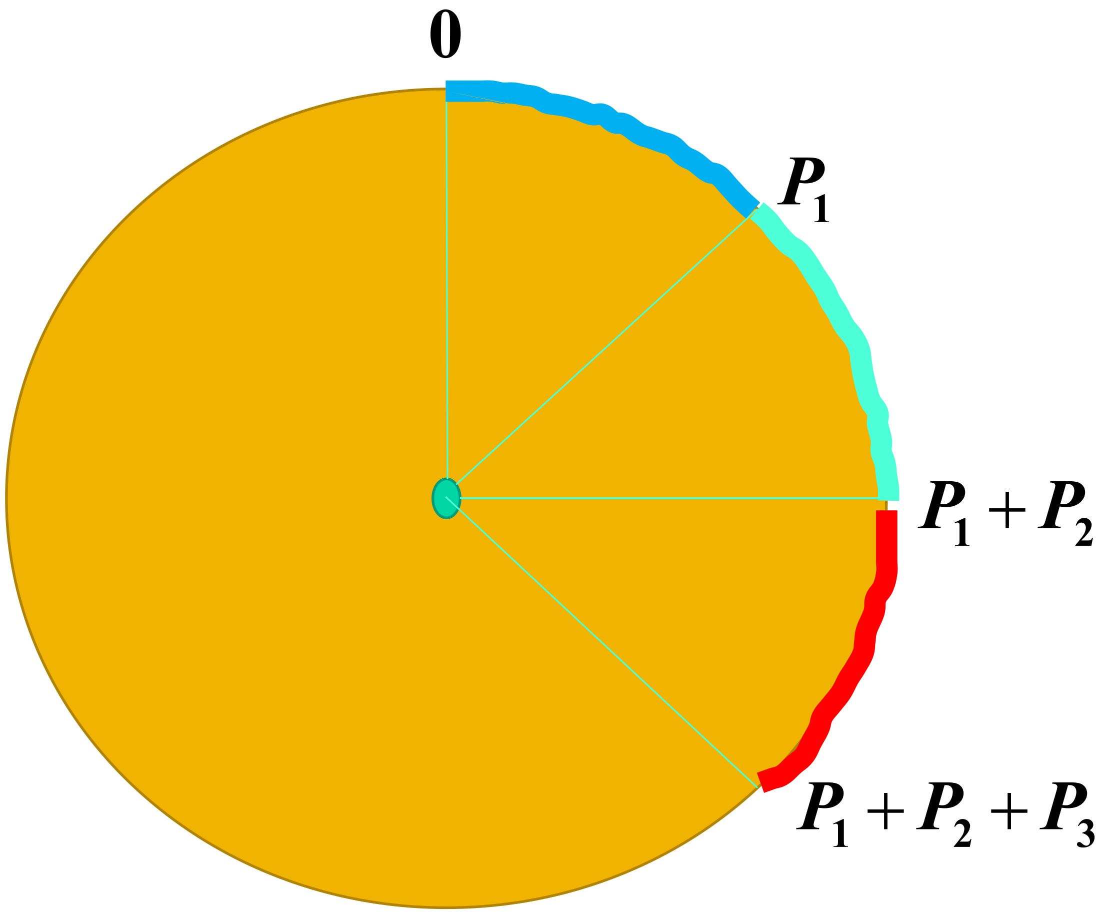
①计算所有个体的适应度值

②计算个体被选中遗传到下一代群体的概率

③模拟赌盘操作来确定个体是否遗传到下一代

$$P_i = F_i / \sum_{i=1}^{n} F_i$$

模拟轮盘赌：



$$F_i=\frac{1}{Obj((p_1,p_2,\ldots,p_n))}$$

$$P_i=F_i/\sum_{i=1}^{n}F_i$$

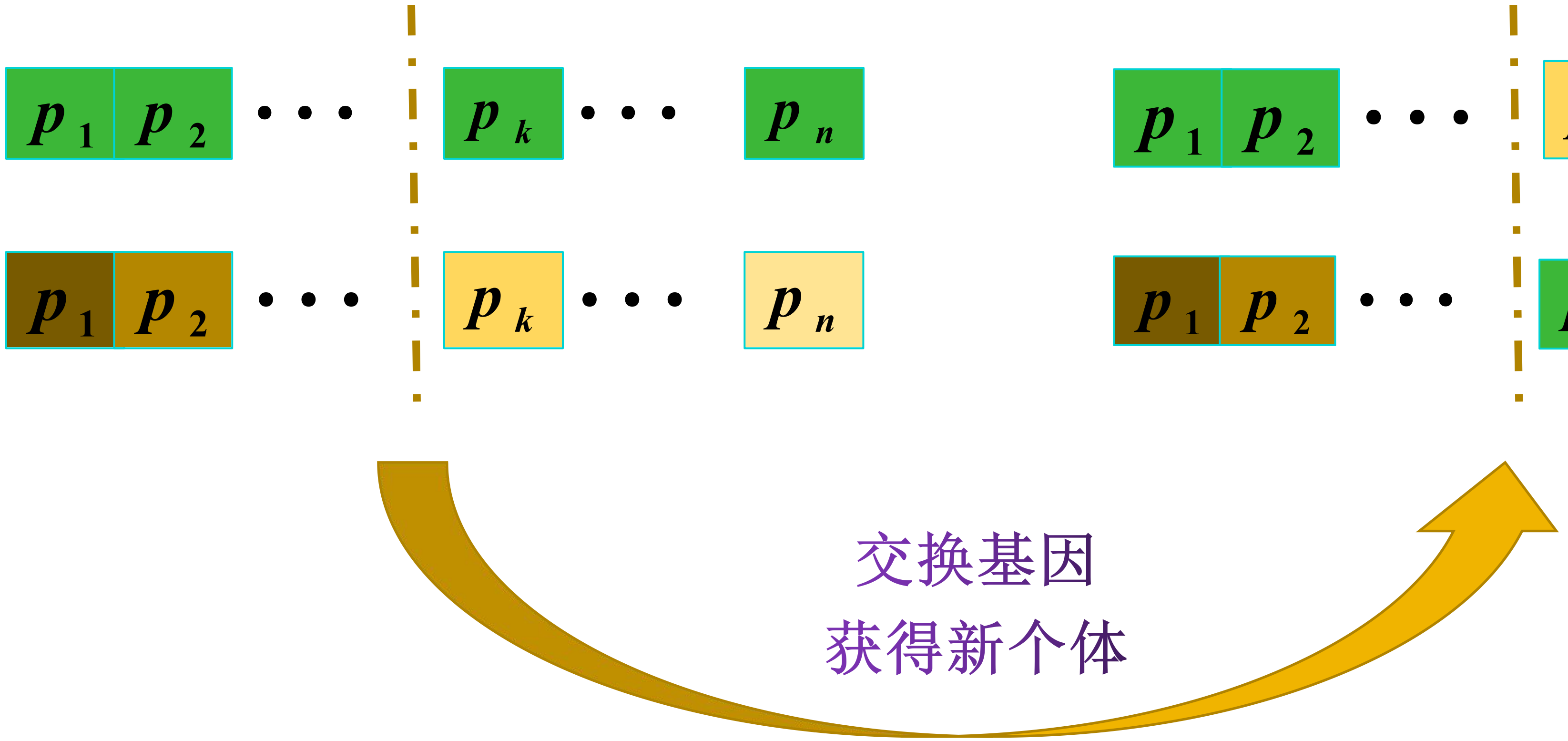
✓产生一个0到1之间的随机数

✓该随机数出现在哪一个区域内

✓确定个体是否被选则

4 群体转移：交叉算子

- 两个配对染色体依据交叉概率 Pc 交换其部分基因，从而形成两个新的个体。



例如，二进制编码染色体的交叉

交叉前(用”|”来表示交叉点)：

00000|01110000000010000

11100|00000111111000101

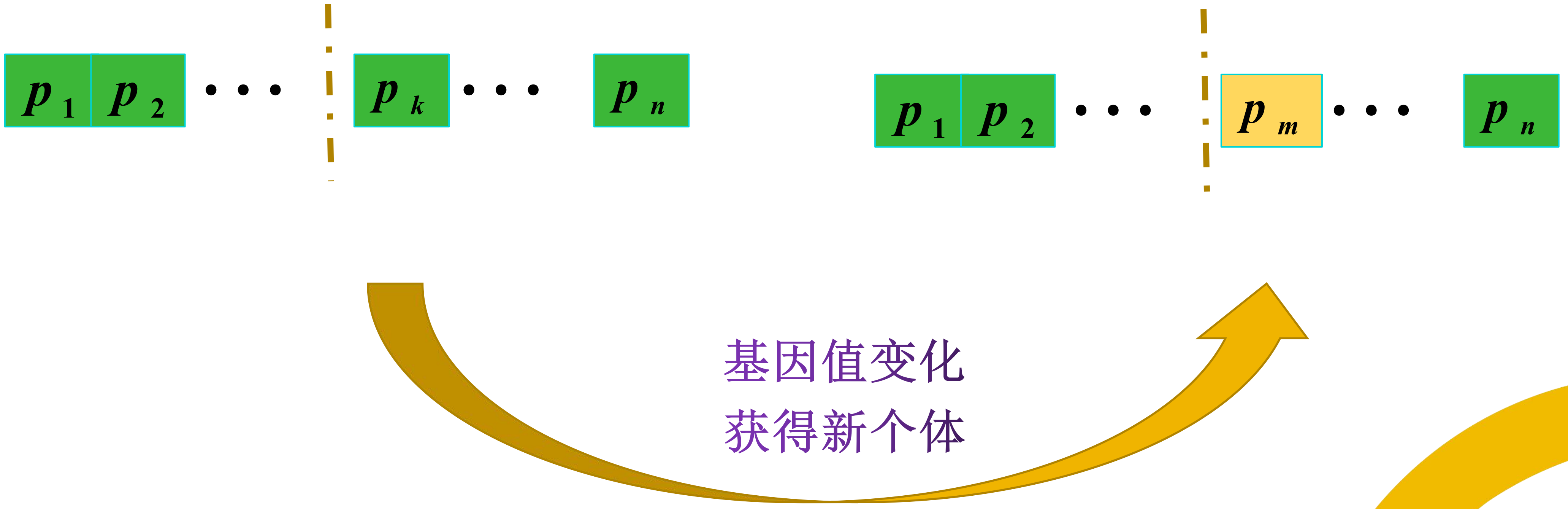
交叉后：

00000|00000111111000101

11100|01110000000010000

5 群体转移：变异算子

- 依据变异概率 Pm 将个体编码串中的某些基因值用其它基因值来替换，从而形成一个新的个体。



例如，二进制编码染色体的变异

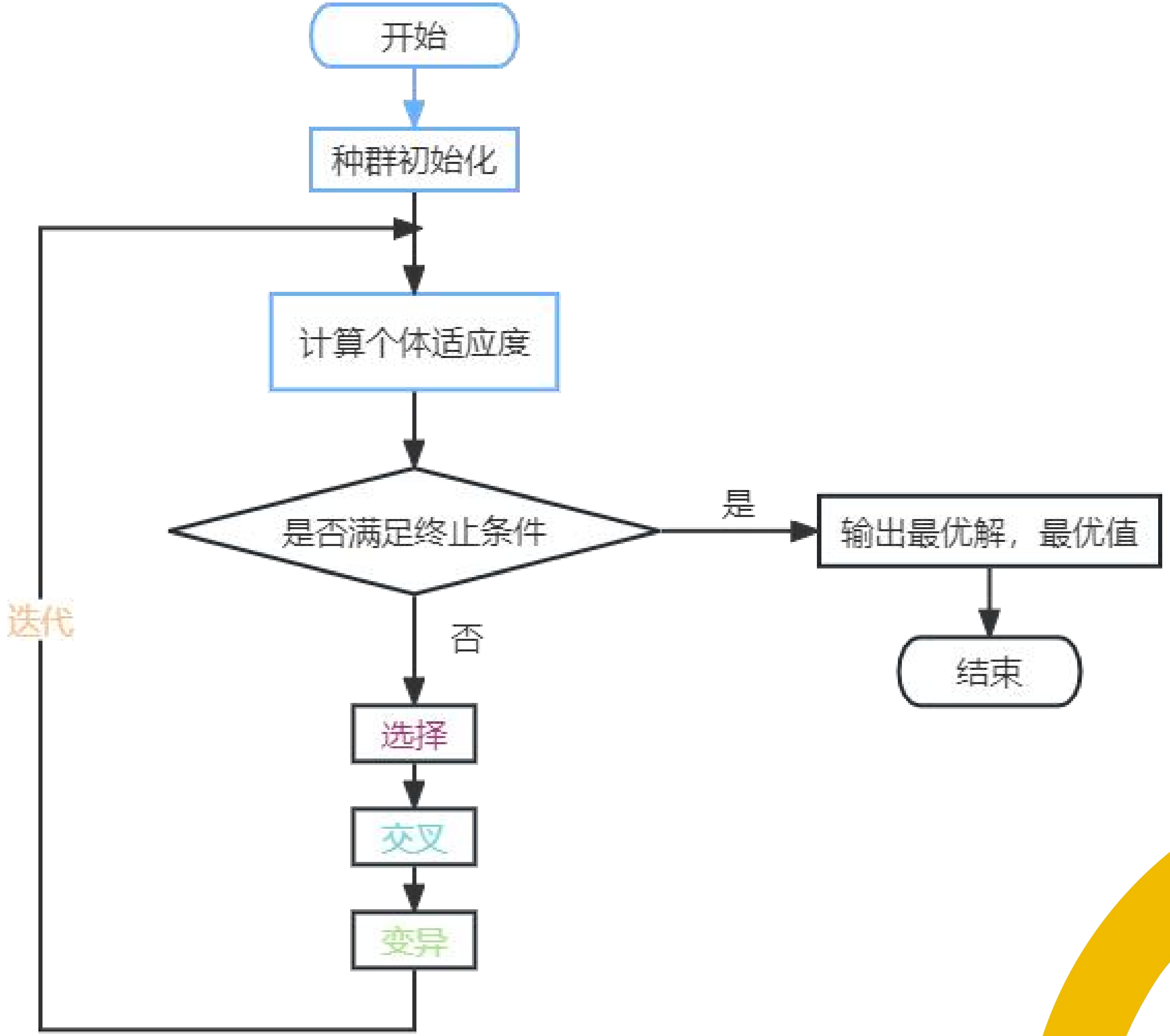
变异前(用”|”来表示变异点)：

11100|00000111111000101

变异后：

11100|10000111111000101

6 算法流程



7 举例

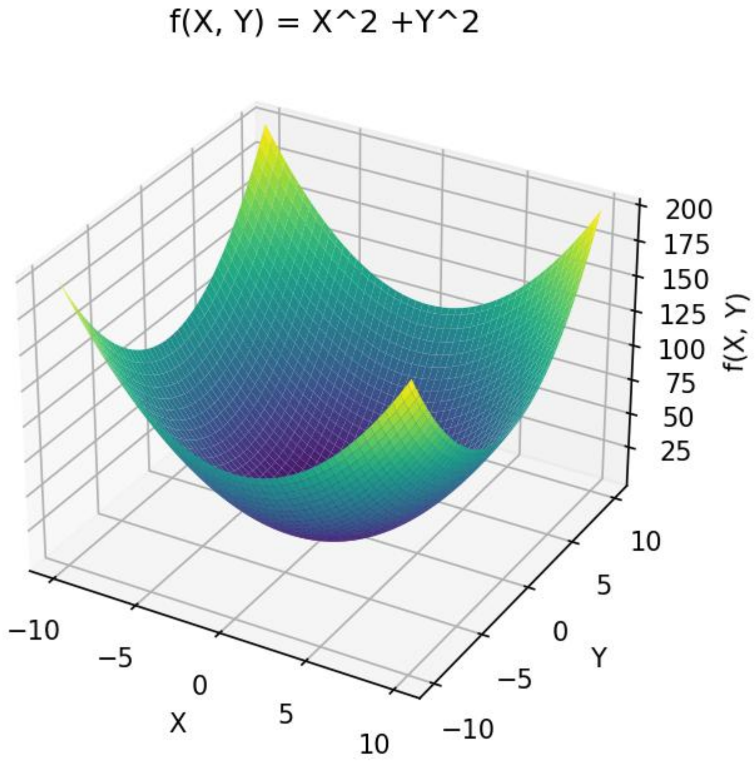
$$f(x,y) = x^2 + y^2$$

利用遗传算法迭代求解最小值

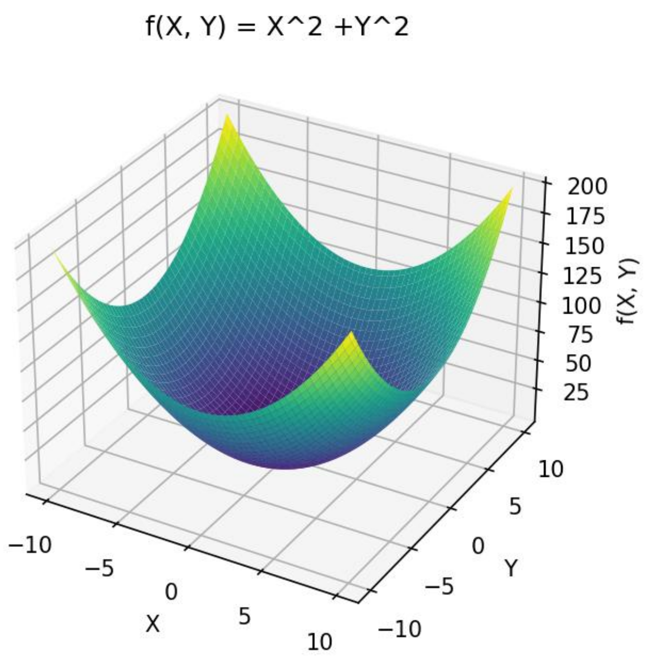
分析

```python
import random
import numpy as np
import matplotlib.pyplot as plt

# 定义函数 f
def f(x, y):
    return x**2 + y**2
```



```python
# 执行遗传算法
# 生成网格点数据
x = np.linspace(-10, 10, 100)
y = np.linspace(-10, 10, 100)
X, Y = np.meshgrid(x, y)
# 计算函数值
Z = f(X, Y)
# 绘制函数图像
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis')
# 设置坐标轴标签
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('f(X, Y)')
# 设置图像标题
plt.title('f(X, Y) = X^2 +Y^2')
# 显示图像
plt.show()
```



```python
# 遗传算法参数设置
population_size = 50  # 种群大小
chromosome_length = 2  # 染色体长度（二维坐标）
# 初始化种群
1 usage
def create_population():
    population = []
    for _ in range(population_size):
        individual = [random.uniform(-10, 10)
                      for _ in range(chromosome_length)]
        population.append(individual)
    return population
```

```python
# 评估种群中个体的适应度
1 usage
def compute_fitness(population):
    fitness_scores = []
    for individual in population:
        x, y = individual[0], individual[1]
        fitness = 1/(f(x, y)+0.001)
        fitness_scores.append(fitness)
    return fitness_scores
```

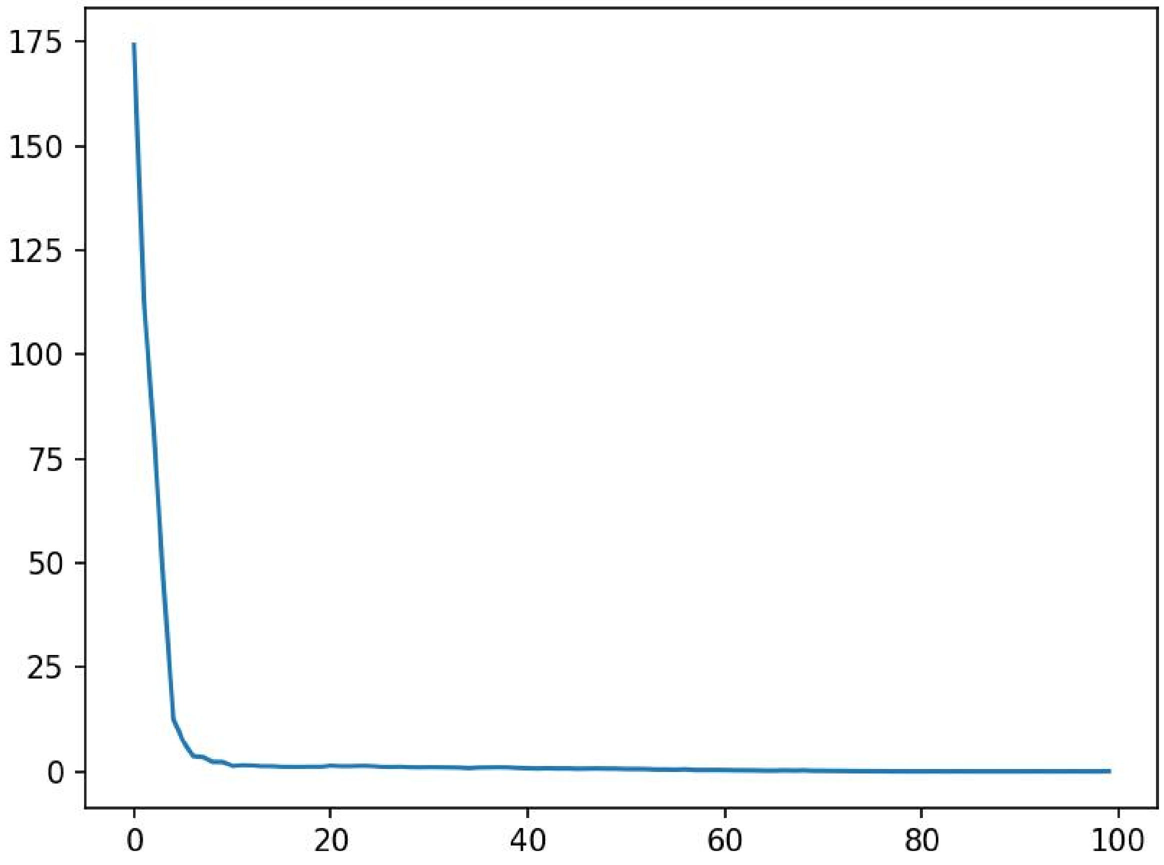
```python
# 选择操作 - 使用轮盘赌选择法
def selection(population, fitness_scores):
    selected_population = []
    total_fitness = sum(fitness_scores)
    while len(selected_population) < population_size:
        rand_num = random.uniform(0, total_fitness)
        cumulative_fitness = 0
        for i, fitness in enumerate(fitness_scores):
            cumulative_fitness += fitness
            if cumulative_fitness > rand_num:
                selected_population.append(population[i])
                break
    return selected_population
```

```python
# 交叉操作 - 使用单点交叉
def crossover(population):
    offspring = []
    for i in range(0, population_size, 2):
        parent1 = population[i]
        parent2 = population[i+1]
        crossover_point = random.randint(1, chromosome_length - 1)
        child1 = parent1[:crossover_point] + parent2[crossover_point:]
        child2 = parent2[:crossover_point] + parent1[crossover_point:]
        offspring.extend([child1, child2])
    return offspring
```

```python
# 变异操作 - 在染色体上进行随机变异
1 usage
def mutation(population):
    mutated_population = []
    for individual in population:
        if random.random() < 0.1:  # 变异概率为0.1
            mutated_individual = []
            for gene in individual:
                new_gene = gene + random.uniform(-0.1, 0.1)
                # 变异范围为[-0.1, 0.1]
                mutated_individual.append(new_gene)
            mutated_population.append(mutated_individual)
        else:
            mutated_population.append(individual)
    return mutated_population
```

```python
# 主函数
1 usage
def genetic_algorithm():
    population = create_population()
    generations = 100  # 迭代次数
    obj_value = []
    for _ in range(generations):
        temp_individual = max(population, key=lambda x: f(x[0], x[1]))
        obj_value.append(f(temp_individual[0], temp_individual[1]))
        fitness_scores = compute_fitness(population)
        selected_population = selection(population, fitness_scores)
        offspring = crossover(selected_population)
        mutated_population = mutation(offspring)
        population = mutated_population
    # 计算最好的个体
    best_individual = max(population, key=lambda x: f(x[0], x[1]))
    best_obj = f(best_individual[0], best_individual[1])
    print("最优解: ", best_individual)
    print("最小值: ", best_obj)
    plt.plot(obj_value)
    plt.show()
```

```
genetic_algorithm()    最优解: [0.05545082313034909, -0.07818789686163227]
                       最小值: 0.009188141001478503
```



模型与代码探索技术：ChatGPT

- ✓ https://chat.gamejx.cn

实例代码：

- ✓ https://github.com/wujian1112/Compute_Intelligence


chato
Chat8 编辑 窗口
New chat
GPT-3.5
GPT-4.0
1天前
搜索最优解…
一周前
在一个图或…
30天前
如何实现遗…
你儿
generate a g…
你结婚了吗
开通VIP
分享好友
best_chromosome, best_fitness = genetic_algorithm()
x_best, y_best = best_chromosome
print("最优解：x =", x_best, "y =", y_best)
print("最大值：", best_fitness)
这段代码实现了基本的遗传算法来求解函数 f(x, y) = -x^2 - y^2 的最大值。遗传算法的步骤包括种群初始化、适应度计算、选择操作（轮盘赌选择）、交叉操作（单点交叉）和变异操作。通过迭代执行这些步骤，不断优化种群中的染色体，最终获得最优解和最大值。
代码中设置了种群大小、染色体长度、变异率和最大迭代次数等参数。你可以根据需要自行调整这些参数。运行代码后，程序将输出求解的最优解和最大值。请注意，由于遗传算法的随机性质，每次运行可能得到不同的结果。
遗传算法求解函数 f(x, y) = -x^2 - y^2 的最大值的Python代码
43/2000
语音识别
GPT-3.5

参考文献

[1] https://blog.csdn.net/qq_37587824/article/details/104110006

[2] 范旭、陈克伟、魏曙光。 Python 智能优化算法（从原理到代码实现与应用），电子工业出版社，北京，2022.

[3] 郁磊、史峰、王辉、胡斐.Matlab 智能优化算法30个案例分析. 北京航空航天大学出版社，2015.

谢谢