第 7 章

1. 试用 OpenSeespy 求解包含 10 根杆件的桁架结构，材料的弹性模量为 20000 MPa，截面面积为 15 mm^2，如图 7-6 所示。

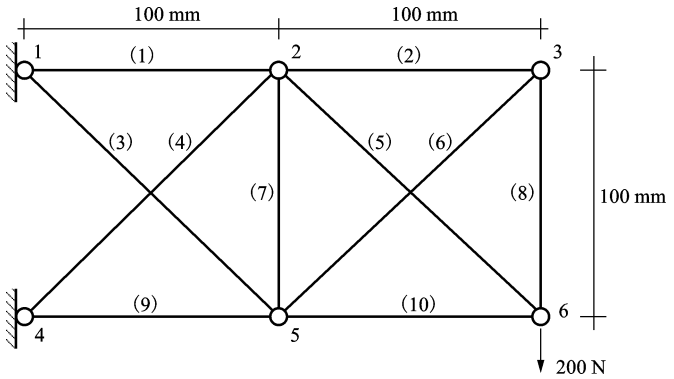

图 7-6　包含 10 根杆件的桁架结构

答案：

```
1.  import openseespy.opensees as ops
2.  import opsvis as ovs
3.  import matplotlib.pyplot as plt
4.
5.  #建立一个桁架，可以改变截面面积和边界
6.  class Truss:
7.      def __init__(self, Areas):
8.          self.Areas=Areas# 所有杆件的截面面积
9.
10.     def Model(self, Areas):
11.         """
12.         采用 openseespy 对桁架进行建模
13.         : param Areas: 桁架截面面积
14.         : return:
15.         """
16.         ops.wipe()
17.         ops.model('basic', '-ndm', 2, '-ndf', 2)  # 平面桁架模型，2 维空间，2 个自由度
18.
19.         # 创建节点
20.         ops.node(1, 0.0, 100.0)# 节点编号，x 坐标，y 坐标
21.         ops.node(2, 100.0, 100.0)
22.         ops.node(3, 200.0, 100.0)
```

```
23.         ops.node(4，0.0，0.0)
24.         ops.node(5，100.0，0.0)
25.         ops.node(6，200.0，0.0)
26.
27.         # 设置边界条件
28.         ops.fix(1，1，1)# 节点编号，是否限值 x 向自由度，是否限值 y 向自由度，1 表示限制，0 表示释放
29.         ops.fix(4，1，1)
30.
31.         # 设置材料属性
32.         ops.uniaxialMaterial("Elastic"，1，20000.0)  # 1 是材料编号，20000 为弹性模量
33.
34.         # 创建单元
35.         ops.element("Truss"，1，1，2，Areas[0]，1)  # 单元类型，单元编号，单元起点，单元终点，单元面
积，材料属性
36.         ops.element("Truss"，2，2，3，Areas[1]，1)
37.         ops.element("Truss"，3，1，5，Areas[2]，1)
38.         ops.element("Truss"，4，4，2，Areas[3]，1)
39.         ops.element("Truss"，5，2，6，Areas[4]，1)
40.         ops.element("Truss"，6，5，3，Areas[5]，1)
41.         ops.element("Truss"，7，5，2，Areas[6]，1)
42.         ops.element("Truss"，8，6，3，Areas[7]，1)
43.         ops.element("Truss"，9，4，5，Areas[8]，1)
44.         ops.element("Truss"，10，5，6，Areas[9]，1)
45.
46.         # 创建加载方法
47.         ops.timeSeries("Linear"，1)  # 采用线性增加的方法，编号为 1
48.         ops.pattern("Plain"，1，1)  # 加载模式
49.
50.         # 施加荷载
51.         ops.load(6，0.0，- 200.0)
52.
53.         # 可视化模型
54.         ovs.plot_model()
55.         # plt.show()
56.
57.       def run(self):
58.         """计算"""
59.         self.Model(self.Areas)
60.         ops.system("BandSPD")
61.         ops.numberer("RCM")
62.         ops.constraints("Plain")
63.         ops.integrator("LoadControl"，1)  # 弹性问题，只加载一步
64.         ops.algorithm("Linear")
65.         ops.analysis("Static")
66.         ops.analyze(1)
67.
68.         # 输出结果
```

```
        ux=ops.nodeDisp(6, 1)# 6 号节点 x 方向位移
        uy=ops.nodeDisp(6, 2)# 6 号节点 x 方向位移

        return ux, uy


if __name__=='__main__':
    Areas=[15.]* 10# 10 根杆件的面积
    trussmodel=Truss(Areas)# 创建分析模型
    UX, UY=trussmodel.run()# 运行

    # 输出
    print(f"当面积为{Areas}时, 6 号节点的 X, Y 向位移分别为: UX={UX}, UY={UY}")
```

X 向位移为-0.136 mm，Y 向位移为-0.537 mm。

2. 试简述遗传算法中选择、交叉和变异算子的功能。

答案：

选择的作用是筛选优秀个体，保留优良基因，淘汰较差的个体；交叉用于重组基因，生成新的解；变异用于引入随机性，避免早熟收敛。

3. 遗传算法一般对多个目标进行优化时可采用罚函数法把多个目标转换成一个目标。例如，对于思考题 1 中的桁架算例，假定桁架截面面积为 2 ~ 30 mm^2，杆件的密度为 0.1 kg/mm^3，6 号节点的竖向位移为 y_1，桁架质量为 y_2，试采用遗传算法工具箱 geatpy 对截面尺寸进行优化，使得 $F=y_1/0.759+0.5y_2/3497$ 最小。

答案：

文件 1：TrussModel.py

```
import openseespy.opensees as ops
import opsvis as ovs
import matplotlib.pyplot as plt

#建立一个桁架, 可以改变截面面积和边界
class Truss:
    def __init__(self, Areas):
        self.Areas=Areas# 所有杆件的截面面积

    def Model(self, Areas):
        """
        采用 openseespy 对桁架进行建模
        : param Areas: 桁架截面面积
        : return:
        """
        ops.wipe()
        ops.model('basic', '-ndm', 2, '-ndf', 2)  # 平面桁架模型, 2 维空间, 2 个自由度

        # 创建节点
        ops.node(1, 0.0, 100.0)# 节点编号, x 坐标, y 坐标
```

```
21.             ops.node(2，100.0，100.0)
22.             ops.node(3，200.0，100.0)
23.             ops.node(4，0.0，0.0)
24.             ops.node(5，100.0，0.0)
25.             ops.node(6，200.0，0.0)
26.
27.             # 设置边界条件
28.             ops.fix(1，1，1)# 节点编号，是否限值 x 向自由度，是否限值 y 向自由度，1 表示限制，0 表示释放
29.             ops.fix(4，1，1)
30.
31.             # 设置材料属性
32.             ops.uniaxialMaterial("Elastic"，1，20000)  # 1 是材料编号，3000 为弹性模量
33.
34.             # 创建单元
35.             ops.element("Truss"，1，1，2，Areas[0]，1)  # 单元类型，单元编号，单元起点，单元终点，单元
面积，材料属性
36.             ops.element("Truss"，2，2，3，Areas[1]，1)
37.             ops.element("Truss"，3，1，5，Areas[2]，1)
38.             ops.element("Truss"，4，4，2，Areas[3]，1)
39.             ops.element("Truss"，5，2，6，Areas[4]，1)
40.             ops.element("Truss"，6，5，3，Areas[5]，1)
41.             ops.element("Truss"，7，5，2，Areas[6]，1)
42.             ops.element("Truss"，8，6，3，Areas[7]，1)
43.             ops.element("Truss"，9，4，5，Areas[8]，1)
44.             ops.element("Truss"，10，5，6，Areas[9]，1)
45.
46.             # 创建加载方法
47.             ops.timeSeries("Linear"，1)  # 采用线性增加的方法，编号为 1
48.             ops.pattern("Plain"，1，1)  # 加载模式
49.
50.             # 施加荷载
51.             ops.load(6，0.，- 200.)
52.
53.         def run(self):
54.             """计算"""
55.             self.Model(self.Areas)
56.             ops.system("BandGeneral")
57.             ops.numberer("Plain")
58.             ops.constraints("Plain")
59.             ops.integrator("LoadControl"，1)  # 弹性问题，只加载一步
60.             ops.algorithm("Newton")
61.             ops.analysis("Static")
62.             ops.analyze(1)
63.
64.             # 输出结果
65.             uy＝ops.nodeDisp(6，2)# 6 号节点 y 方向位移
66.             return uy
```

```
67.
68.
69. if __name__=='__main__':
70.     Areas=[15.]* 10
71.     trussmodel=Truss(Areas)# 创建分析模型
72.     UY=trussmodel.run()# 运行
73.     ovs.plot_model()
74.     ovs.plot_loads_2d()
75.     plt.show()
76.
77.     # 输出
78.     print(f"当面积为{sum(Areas)}时，6 号节点的 Y 向位移为：UY={UY}")
```

文件 2：GA. py

```
1.  import numpy as np
2.  import geatpy as ea
3.  import time
4.  from TrussModel import Truss
5.
6.  """==========================目标函数=========================
========"""
7.
8.  def aim(Phen):    # 传入种群染色体矩阵解码后的基因表现型矩阵，该矩阵的行数为个体数目 N，列数为
每个染色体维度，这里是桁架面积
9.      OBjv=np.zeros((len(Phen)，1))# 初始化目标函数值矩阵，是一个列向量，行数为染色体个数
10.     for i in range(len(Phen)):
11.         # 提取每个个体的表现型，让总面积不超过 30* 31
12.         Areai=Phen[i]* 30.* 10. / sum(Phen[i])
13.         Areai=np.clip(Areai，2.，30.)
14.         # 创建 opensees 模型
15.         truss=Truss(Areai)
16.         uy=truss.run()# 计算 6 号节点的竖向位移
17.         # 计算桁架质量
18.         # 长度矩阵
19.         lx=np.sqrt(100.* * 2+100.* * 2)# 斜杆的长度
20.         L=np.array([100.，100.，lx，lx，lx，lx，100.，100.，100.，100.])
21.         m=sum(0.1* L* Areai)
22.
23.         OBjv[i，0]=abs(uy)/0.759+0.5* m/3497.# 把计算结果赋予目标函数值矩阵，这里要把位移变成正值
            24.return OBjv
25.
26.
27.     if __name__=='__main__':
28.         """=========================变量设置=================
==========="""
29.         DIM=10# 决策变量的维度，这里有 10 个桁架，即 10 个桁架的面积，所以维度是 10
30.         X=[2.，30.]# 决策变量的范围，
31.         B=[1，1]# 决策变量的边界，1 表示包含范围的边界，0 表示不包含
```

```
        # 生成自变量的范围矩阵，每一列对应一个决策变量的下界和上界
        ranges=np.array([X]* DIM).T
        # 生成自变量的边界矩阵，每一列对应一个决策变量是否取到下界和上界
        borders=np.array([B]* DIM).T
        varTypes=np.array([0]* DIM)# 决策变量的类型，0 表示连续，1 表示离散

        """==========================染色体编码设置==========================="""
        Encoding=' BG'    # ' BG' 表示采用二进制/格雷编码
        codes=[1]* DIM# 决策变量的编码方式，DIM 个 1 表示变量均使用格雷编码
        precisions=[2]* DIM# 决策变量的编码精度，表示解码后能表示的决策变量的精度可达到小数点后 6 位
        scales=[0]* DIM# 0 表示采用算术刻度，1 表示采用对数刻度#调用函数创建译码矩阵
        # 生成译码矩阵，该矩阵实现表现型和基因型的转换
        FieldD=ea.crtfld(Encoding，varTypes，ranges，borders，precisions，codes，scales)
        # FieldD=ea.crtfld(Encoding，varTypes，ranges，borders)

        """==========================遗传算法参数设置==========================="""
        NIND=200# 种群个体数目
        MAXGEN=200# 最大遗传代数
        maxormins=np.array([1])# 表示目标函数是最小化，元素为-1 则表示对应的目标函数是最大化
        selectStyle=' sus'    # 采用随机抽样选择
        recStyle=' xovdp'    # 采用两点交叉
        mutStyle=' mutbin'    # 采用二进制染色体的变异算子
        Lind=int(np.sum(FieldD[0，：]))# 计算染色体长度
        pc=0.9# 交叉概率
        pm=1./ Lind# 变异概率
        obj_trace=np.zeros((MAXGEN，2))# 定义目标函数记录器
        var_trace=np.zeros((MAXGEN，Lind))# 染色体记录器，记录历代最优个体的染色体

        """==========================开始遗传算法进化==========================="""
        start_time=time.time()# 开始计时
        # 初代种群生成
        Chrom=ea.crtpc(Encoding，NIND，FieldD)# 生成种群染色体矩阵
        Phen=ea.bs2ri(Chrom，FieldD)# 对初始种群进行解码
        ObjV=aim(Phen)# 计算初始种群个体的目标函数值
        # best_ind=np.argmin(ObjV)    # 计算当代最优个体的序号
        FitnV=ea.ranking(ObjV* maxormins)
        best_ind=np.argmax(FitnV)# 计算当代最优个体的序号

        # 开始进化
        for gen in range(MAXGEN):
            FitnV=ea.ranking(maxormins* ObjV)# 根据目标函数大小分配适应度值
            SelCh=Chrom[ea.selecting(selectStyle，FitnV，NIND-1)，:]# 选择
            SelCh=ea.recombin(recStyle，SelCh，pc)# 重组
```

```
            SelCh=ea.mutate(mutStyle, Encoding, SelCh, pm)# 变异
            # #把父代精英个体与子代的染色体进行合并，得到新一代种群
            Chrom=np.vstack([Chrom[best_ind, :], SelCh])
            Phen=ea.bs2ri(Chrom, FieldD)# 对种群进行解码(二进制转十进制)
            ObjV=aim(Phen)# 求种群个体的目标函数值
            # 记录
            FitnV=ea.ranking(maxormins* ObjV)
            best_ind=np.argmax(FitnV)# 计算当代最优个体的序号
            obj_trace[gen, 0]=np.sum(ObjV) / ObjV.shape[0]# 记录当代种群的目标函数均值
            obj_trace[gen, 1]=ObjV[best_ind]# 记录当代种群最优个体目标函数值
            var_trace[gen, :]=Chrom[best_ind, :]# 记录当代种群最优个体的染色体
            print(f"目前第{gen}次迭代，最优目标函数值为{ObjV[best_ind]}")
            # 进化完成
        end_time=time.time()# 结束计时
        ea.trcplot(obj_trace, [['种群个体平均目标函数值', '种群最优个体目标函数值']])  # 绘制图像

        """==========================输出结果==========================="""

        best_gen=np.argmin(maxormins* obj_trace[:, [1]])
        print('最优解的目标函数值: ', obj_trace[best_gen, 1])
        variable=ea.bs2ri(var_trace[[best_gen], : ], FieldD)# 解码得到表现型(即对应的决策变量值)
        variable=variable* 30.* 10. / np.sum(variable)
        variable=np.clip(variable, 2., 30.)
        print('最优解的决策变量值为: ')
        for i in range(variable.shape[1]):
            print('x' +str(i)+' =', variable[0, i])
        print('用时: ', end_time - start_time, '秒')
        np.savetxt('results.txt', obj_trace)
```