```python
# 各型号车辆信息表
table1 = {...}
# 运单信息表
table2 = {...}
# 故障信息表
table3 = {...}

pdTable1 = pd.DataFrame(table1)
pdTable2 = pd.DataFrame(table2)
pdTable3 = pd.DataFrame(table3)

# 单位中转成本
def e(m, n):...


# 单位中转时间
def A(m, n):...
```

```python
# 转运成本计算
def C1(Table2=pdTable2):...


# 运输成本
def C2(Table2=pdTable2):...


# 运输路程时间
def T0(X, Table1=pdTable1, Table2=pdTable2):...


# 物料转运时间  A为中转用时，默认0.5
def T1(X,Table2=pdTable2, A=0.5):...

# 迎接集中停运时间
def T2(X, C=3):...

# 载具故障维修所需时间
def T3(X, Table1=pdTable1, Table3=pdTable3):...
```

```python
# 时间成本
def C3(X):...


# 优化函数
def Fun(X):...


# 随机解X  将批量展开，中转后的车展开，得到38维度
def initialX(Table1=pdTable1, Table2=pdTable2):...


# 约束条件
def constraints(X, Table1=pdTable1, Table2=pdTable2):...
```

```
#
# X = initialX()
# while constraints(X) == False:
#     X = initialX()
# X = [31,32,32,32,33,  # 31运订单1    32，32运订单2（两个批量）    32运订单3    33运订单4
#      34,34,35,21,21,  # 34运订单5    34运订单6      35运订单7    21运订单8     21运订单9
#      0,1,2,2,2,
#      2,2,2,2,3,
#      19,3,3,3,3,
#      3,3,3,3,3,
#      3,3,3,4,22,
#      22,36,36]

# 例X 中 前34个为运输第一个地点，后4个为运输中转后第二个地点
```

```
'''优化函数'''
#,y = x^2     用户可以自己定义其他函数
Y1=[10,3,15,2]
def fun(X):
    output = (sum(np.square(X-Y1))/len(X))**(1/2)
    return output


''' 种群初始化函数 '''


def initial(pop, dim):
    X = [[0]*dim]*pop
    for i in range(pop):
        X[i],lb,ub=initialX()
        while constraints(X[i]) == False:
            X[i],lb,ub = initialX()

    return X, lb, ub
```

```
'''边界检查函数'''


def BorderCheck(X, Ub, Lb, pop, dim):
    for i in range(pop):
        for j in range(dim):
            ub = Ub[j]
            lb = Lb[j]
            x = X[i][j]
            if (X[i][j] >= ub) or (X[i][j] < lb):
                X[i], Ub, Lb = initialX()
                break
        while constraints(X[i]) == False:
            X[i], Ub, Lb = initialX()
    return X


'''计算适应度函数'''
```

```python
def CaculateFitness(pop,X, fun):
    fitness = [0]*pop
    for i in range(pop):
        fitness[i] = fun(X[i])
    return fitness


'''适应度排序'''


def SortFitness(Fit):
    fitness = list(np.sort(Fit, axis=0))
    index = list(np.argsort(Fit, axis=0))
    return fitness, index
```

```python
'''根据适应度对位置进行排序'''
def SortPosition(pop, dim,X, index):
    Xnew = [[0]*dim]*pop
    for i in range(pop):
        Xnew[i] = X[index[i]]
    return Xnew
```

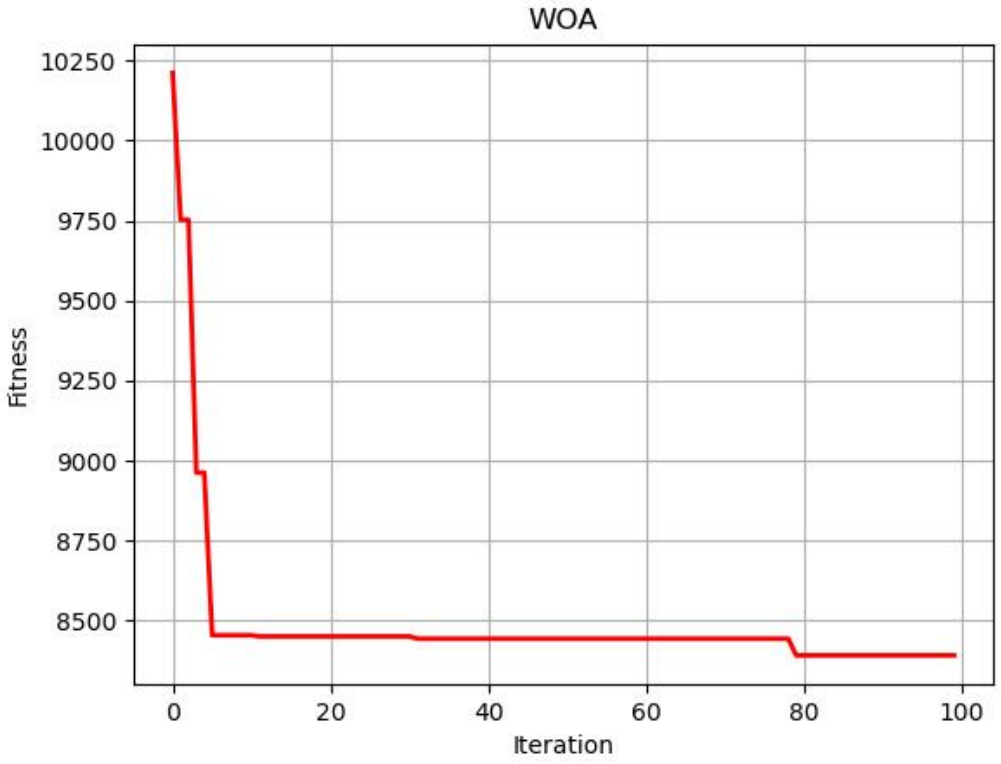
```python
'''鲸鱼优化算法'''
def WOA(pop, dim,MaxIter, fun):
    X, lb, ub = initial(pop, dim)  # 初始化种群
    fitness = CaculateFitness(pop,X, fun)  # 计算适应度值
    fitness, sortIndex = SortFitness(fitness)  # 对适应度值排序
    X = SortPosition(pop, dim,X, sortIndex)  # 种群排序
    GbestScore = fitness[0]
    GbestPositon = np.zeros([1, dim])
    GbestPositon[0, :] = X[0]
    Curve = np.zeros([MaxIter, 1])
    for t in range(MaxIter):...
    return GbestScore, GbestPositon, Curve
```

```python
'''主函数 '''
# 设置参数
pop = 20  # 种群数量
MaxIter = 100  # 最大迭代次数
P1 = np.cumsum(list(pdTable2["批量"]))
P2 = np.cumsum(np.multiply(list(pdTable2["中转地点"]), list(pdTable2["批量"])))
dim = P1[-1] + P2[-1]
# dim = 4  # 维度

GbestScore, GbestPositon, Curve = WOA(pop, dim, MaxIter, Fun)
print('最优适应度值：', GbestScore)
print('最优解：', GbestPositon)

# 绘制适应度曲线
plt.figure(1)
plt.plot(Curve, 'r-', linewidth=2)
plt.xlabel('Iteration', fontsize='medium')
plt.ylabel("Fitness", fontsize='medium')
plt.grid()
plt.title('WOA', fontsize='large')
plt.show()
```

结果例




```
最优适应度值： 8392.255015394912
最优解： [[32. 38. 38. 38. 34. 39. 39. 36. 30. 30.  9.  4. 16. 16. 16. 16. 16. 16.
  16. 20. 20. 20. 19. 19. 19. 19. 19. 19. 19. 19. 19. 19.  0. 29. 29.
  33. 33.]]
```