基于 WOA-Elm 的青铜产品投资策略研究

摘　要

青铜时代是以使用青铜器为标志的人类物质文化发展阶段。青铜是红铜(纯铜)与锡或铅的合金，因埋在土里后颜色因氧化而青灰，故名青铜。青铜可以制作生活、祭祀用器、物件，也可以制作兵器。本文正是基于该问题背景下，研究如何构建规划模型使得投资青铜产品能过使得收益金额最高。

针对问题一, 本文认为是一个典型的路由网络选择问题。对于这种问题而言，首先需要明确各个路网节点(部落)的距离以及各个部落的人数。通过对附件 1 的数据进行分析可知，部落之间的颜色不同并不会对实际情况造成影响。因此，本文首先对上述内容的路网矩阵进行构建，由于不存在路径上的往返方向，因此构建成无向图形式进行求解。对于上述无向图，通过分析每个部落的需求量函数可知，需求量与价格成反比。因此，通过设定不同的参数 c 从而对上述方式进行求解，采用的方法为 **WOA-Elm**，从而得到**最优的青铜器生产投资策略，结论为 C 值取为 0.1 时所得到的利润最大为 28500.**

针对问题二，在问题一的基础上进行分析可知，问题一和问题二的区别在于是否发动战争。根据你提供的新条件，我们将讨论参数 C 的不同取值范围，并制定相应的投资选点、生产、运输和定价计划，以实现某个国家最快打败其他两个国家并实现全部统一的目标。因此，在问题一的基础上，本文将引入双方部落是否会发动战争以及发动战争之后的各青铜器需求量变化，从而对问题一的模型进行修正得到最终的结果。

关键词:WOA-Elm;青铜产品;投资策略;策略研究

一、问题的背景与重述

1.1 问题的背景

青铜时代处于铜石并用时代之后，早于铁器时代之前，在世界范围内的编年范围大约从公元前 4000 年至公元初年。世界各地进入这一时代的年代有早有晚。伊朗南部、美索不达米亚一带在公元前 4000~前 3000 年已使用青铜器欧洲在公元前 4000~前 3000 年、印度和埃及在公元前 3000 前 2000 年，也有了青铜器。埃及、北非以外的非洲使用青铜较晚，大约不晚于公元前 1000 年公元初年。美洲直到将近公元 11 世纪，才出现冶铜中心。在青铜时代，世界上青铜铸造业形成几个重要的地区，这些地区成了人类古代文明形成的中心。青铜器在人们的生产、生活中占据重要地位。

青铜时代是以使用青铜器为标志的人类物质文化发展阶段。青铜是红铜(纯铜)与锡或铅的合金，因埋在土里后颜色因氧化而青灰，故名青铜。青铜可以制作生活、祭祀用器、物件，也可以制作兵器。自有了青铜器和随之的增加，农业和手工业的生产力水平提高，物质生活条件也渐渐丰富，对提高社会生产力起了划时代的作用，同时各地也出现了为青铜而引发的战争，即所谓的青铜之战。“图 1 是青铜时代区域人口分布图，每个方形代表一个部落，里面的数字代表该部落人口《千人)，部落与部落之间的黑线代表有道路连接。蓝色区域属于“沧海王国”，绿色区域属于“桑海王国”，红色区域属于“加纳王国”。

图 1-1 部落分布图

1.2 问题的重述

基于上述问题背景，假定你是一个青铜时代的具有青铜冶炼技术的人，想生产并售卖青铜器(为简单，所有青铜产品不区分产品类型，单位统一为件)赚钱。你想不断选择部落作为青铜器生产点，每个生产点的初始投资费用为 2000 钱币，建设时间需要 1 个季度，从下一个季度开始每季度每个生产点最多能制造出 100 件青铜器，每件青铜器的制造成本为 5 钱币。

你可以对青铜器统一定价 P，一旦定了就不可改变，且每个部落每季度的最大需求量为:

$$C \times \frac{\text{人口(千)}}{\text{价格 P}}$$

只要部落与部落之间有路，每段路每件青铜器配送成本为 1 钱币。当路的两边为不同国家时，这段路每件青铜器的配送成本增加 1 钱币。

1、初始你有现金 6000 钱币。讨论参数 C 的不同取值范围与你相对应的投资选点、生产、运输、定价计划，使得 3 年后最终钱币最多。(假设国家间的地盘没有发生变化，且钱币统一管理)

2、如果相邻 2 个对立部落(属于不同国家且有路相连)，一个部落的青铜器当季度需求 100%得到满足，另一个部落满足量不大于 50%，则满足需求的部落必然发动战争(否则不会发生战争)，如果对于同一个部落同时有多个相邻部落满足发动条件，则人口多的部落会发动战争。战争会持续 2 个季度，期间 2 部落所属国之间的运输道路会全部中断,这 2 个对立部落的青铜器最大需求量都会增加 50%，这段时间如果发动战争的部落青铜器最大需求量连续得到 100%满足，被攻方的最大需求量满足率连续不大于 50%，则进攻方必然占领被攻方的部落，你在该部落的青铜器生产点也会因战败而导致毁坏。如果不满足条件，则一定打平,地盘不发生变化。假设初始你还是有现金 6000 钱币，讨论 C 的不同取值范围与你相对应的投资选点、生产、运输、定价计划，使得某国打败其他两国最快实现全部统一。

二、问题的分析

2.1 问题一的分析

针对问题一，题目要求初始你有现金 6000 钱币。讨论参数 C 的不同取值范围与你相对应的投资选点、生产、运输、定价计划，使得 3 年后最终钱币最多。(假设国家间的地盘没有发生变化，且钱币统一管理)。

对于该问题而言，本文认为是一个典型的路由网络选择问题。对于这种问题而言，首先需要明确各个路网节点(部落)的距离以及各个部落的人数。通过对附件 1 的数据进行分析可知，部落之间的颜色不同并不会对实际情况造成影响。因此，本文首先对上述内容的路网矩阵进行构建，由于不存在路径上的往返方向，因此构建成无向图形式进行求解。对于上述无向图，通过分析每个部落的需求量函数可知，需求量与价格成反比[1]。

因此，通过设定不同的参数 c 从而对上述方式进行求解，采用的方法为 WOA-Elm，从而得到最优的青铜器生产投资策略。

2.2 问题二的分析

针对问题二，在问题一的基础上进行分析可知，问题一和问题二的区别在于是否发动战争。根据你提供的新条件，我们将讨论参数 C 的不同取值范围，并制定相应的投资选点、生产、运输和定价计划，以实现某个国家最快打败其他两个国家并实现全部统一的目标。因此，在问题一的基础上，本文将引入双方部落是否会发动战争以及发动战争之后的各青铜器需求量变化，从而对问题一的模型进行修正得到最终的结果。

三、模型假设

1. 假设在本文中，各个部落的需求量不会随着时间发生变化。
2. 假设在本文中，各个部落在问题一不会因为需求量发生战争。
3. 假设国家之间的地盘不会发生明显变化，也即位置不会发生变化。

四、符号说明

符号	说明
X	标准数据
μ	数据集的平均值
σ	标准差
M	判断矩阵
Q_i	第 i 个部落的最大生产量
N	部落的数量
K_i	第 i 个部落的人口数量(千人)
I_i	表示第 i 个部落的投资费用
S_i	第 i 个部落是否被选择
C_i	第 i 个部落的成本
D_(i,j)	部落 i 到部落 j 之间的距离
R_(i,j)	部落 i 到部落 j 之间的道路链接变量
$Y(i,j,t)$	表示部落 i 是否在 t 时刻对部落 j 发动进攻

五、模型的建立与求解

5.1 数据的预处理

在进行数学建模的构建之前，最为重要的数据的预处理，数据的预处理奠定了该模型所得到的结果是否正确。因此，为了保证数据的准确性和程序的高效率，本文对附件中所提供的附录数据进行了整体审查，删除了错误明显或信息较少的数据，并对异常数据进行了纠正或删除。针对不同类型的错误的具体处理方法如下：

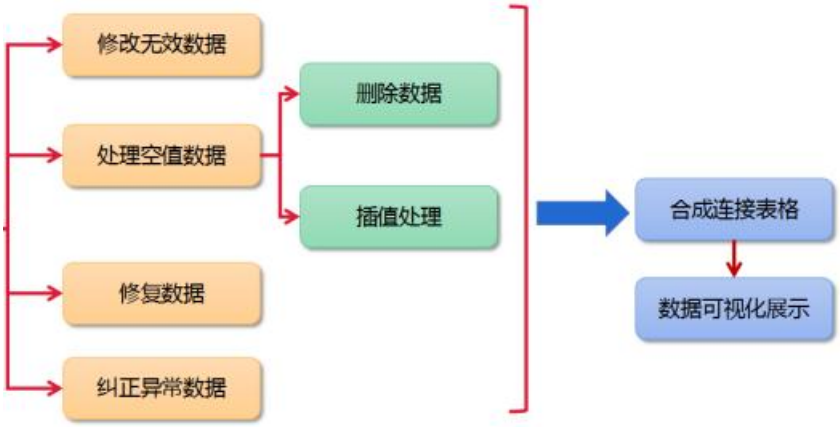


图 5-1　数据预处理流程图

之后，本文将进行归一化处理。因为各个指标的测量单位是不一致的，所以在用这些指标来衡量一个综合指数之前，必须将其归一化，也就是将其绝对的价值转换成相对价值；同时，这样也就是解决了各种指标的同质性问题。另外，正、负指数所表示的意义也是不一样的（正指数值越高，负指数值越低），所以我们采用了不同的方法来进行数据规范化。其具体做法是根据正负指标的不同进行分析：

正向指标:

$$x'_{ij}=\left[\frac{x_{ij}-\min\left(x_{1j},x_{2j},...,x_{nj}\right)}{\max\left(x_{1j},x_{2j},...,x_{nj}\right)-\min\left(x_{1j},x_{2j},...,x_{nj}\right)}\right]$$

负向指标:

$$x'_{ij}=\left[\frac{\max\left(x_{1j},x_{2j},...,x_{nj}\right)-x_{ij}}{\max\left(x_{1j},x_{2j},...,x_{nj}\right)-\min\left(x_{1j},x_{2j},...,x_{nj}\right)}\right]$$

进一步，考虑到各种数据之间存在量纲不一致性，应该先对量纲进行归一化操作。常见的归一化操作如下所示。

1. Z-Score归一化操作

简单来说，Z-Score将数据按比例缩放,使之落入一个特定区间，其具体公式如下：

$$x'=\frac{X-\mu}{\sigma}$$

其中是 X 标准数据， μ 是数据集的平均值， σ 是标准差。

2. Linearnormalization归一化

线性归一化可以说是更容易且更灵活的归一化技术。它通常被称“max-min”归一化，它允许分析人员获取集合中最大x值和最小x值之间的差值，并建立一个基数。这是一个很好的开始策略，实际上，线性归一化可以将数据点归一化为任何基数。下是线性归一化的公式：

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

3. StandardDeviationNormalization归一化

标准差归一化是一种常用的归一化操作。其中归一化值的计算如下所示。

$$Normalized(e_i) = \frac{e_i}{std(E)}$$

当

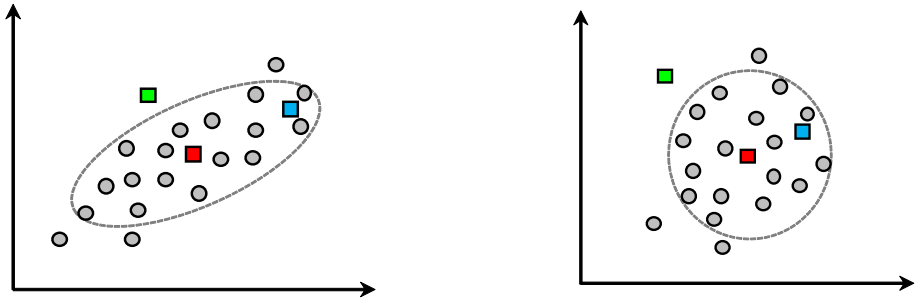
$$std(E) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(e_i - E)^2}$$

则有

$$E = \frac{1}{n}\sum_{i=1}^{n} e_i$$

进行指标数值化和归一化之后，我们使马氏距离模型来进行异常检测，从而剔除掉异常数据。

马氏距离示意图：



(a) (b)

图 5-2 马氏距离和欧几里得距离图

马氏间距与欧几里德距的观点是相反的。用协方差矩阵来描述各个特性的特定参量。可以把直觉的协方差矩阵看作是一个多重正态协方差矩阵。这样，这种分布的密度的曲线就是一个共同的椭圆形。从椭圆形的中央到各个方位的马氏间距都是相同的。图(a)所示的资料点经马氏间距转换，从分布区中央至各点的间距与在附图(b)所示的欧几里德距相等[2-3]。

利用马氏距离，可以将样本中各特征维度之间的关联度和尺度差异进行转化，从而使得新的欧式距离能够有效地测量出样本与分布之间的距离。

在进行指标选取之后，为了进一步确定指标选取的合理性。本文将采用单因素方差分析来判断各个统计量随着时间变化是否存在显著差距。首先，本文对单因素方差分析进行介绍，如下所示

单因素方差分析采用的统计推断方法是计算 F 统计量、进行 F 检验。总的变异平方和记为 SST，分解为两个部分：一部分是由控制变量引起的离差，记为 SSA (组间离差平方和)；另一部分随机变量引起的 SSE(组内离差平方和)。两变量的关系如下所示

$$SST = SSA + SS$$

一般，假设有 k 个水平并建立如下假设：

$$H_0: u_1 = u_2 = \ldots u_k$$

$$H_1: u_1, u_2, \ldots, u_k \text{互不全相等}$$

每个总体系统内部的误差，主要由随机抽样所引起的随机误差，称为组内误差，组内误差平方和记为 SSA：

$$SSA = \sum_{i=1}^{k} n_i (\bar{x_i} - \bar{x})^2$$

各总体之间的误差，主要由随机抽样引起的随机误差和系统误差，称为组间误差，组间误差平方和为 SSE：

$$SSE = \sum_{i=1}^{k} \sum_{k}^{n_i} (x_{ij} - \bar{x_i})^2$$

数据总的误差平方和记为 SST：

$$SST = \sum_{i=1}^{k} \sum_{k}^{n_i} (x_{ij} - \bar{x})^2$$

其中，x_{ij}表示第 i 个水平下第 j 个观测值，$\bar{x_i}$表示第 i 水平的均值，$\bar{x}$表示 n 个观测值的均值。

$$SST = \sum_{i=1}^{k} \sum_{k}^{n_i} (x_{ij} - \bar{x})^2 = SSA + SSE$$

上述计算表明，数据总的误差主要来自两方面：一方面是组内误差，另一方面是组间误差。可以证明，当 k 个总体的均值相等时，$SSA/_{k-1}/SSE/_{n-k}$服从分子自由度为 k-1、分母自由度为 n-k 的 F 分布

$$F = \frac{SSA/(n-1)}{SSE/(n-k)} \leq F_{\alpha}(k-1, n-k)$$

F 值越大，说明在总方差中组间方差波动越大，k 个水平的均值相差越大，越有利于拒绝 k 个总体均值相等的假设；F 值越小，说明总方差中组间方差波动越小，方差的波动主要是由随机误差引起的，越有利于接受 k 个总体相等的假设。如果规定显著性水平为 a，当 p 值小于 a 时，拒绝原假设；当 p 值大于 a 时，接受原假设。

5.2 问题一模型的建立

针对问题一，题目要求初始你有现金 6000 钱币。讨论参数 C 的不同取值范围与你相对应的投资选点、生产、运输、定价计划，使得 3 年后最终钱币最多。(假设国家间的地盘没有发生变化，且钱币统一管理)。

对于该问题而言，本文认为是一个典型的路由网络选择问题。对于这种问题而言，首先需要明确各个路网节点(部落)的距离以及各个部落的人数。通过对附件 1 的数据进行分析可知，部落之间的颜色不同并不会对实际情况造成影响。因此，本文首先对上述内容的路网矩阵进行构建，由于不存在路径上的往返方向，因此构建成无向图形式进行求解。对于上述无向图，通过分析每个部落的需求量

函数可知，需求量与价格成反比[4]。

因此，通过设定不同的参数 c 从而对上述方式进行求解，采用的方法为 WOA-Elm，从而得到最优的青铜器生产投资策略。

首先，本文对参数进行设定，从而表示不同参数与模型之间的关系。C 表示每个部落需求量、人口数量和定价 P 之间的关系。N 表示部落的数量，K_i 表示第 i 个部落的人口数量(千人)，I_i 表示第 i 个部落的投资费用，S_i 表示第 i 个部落是否被选择，则可以表示如下所示[5]

$$S_{\_i}=\begin{cases}S_{\_i}=1,\text{表示该部落选作为生产部落}\\ S_{\_i}=0,\text{表示该部落未被选作为生产部落}\end{cases}$$

Q_i 表示为第 i 个部落的最大生产量，C_i 表示为第 i 个部落的生产成本，D_(i,j)表示为部落 i 到部落 j 之间的距离，R_(i,j)表示为部落 i 到部落 j 之间的道路链接变量，则有

$$R_{\_(i,j)}=\begin{cases}R_{\_(i,j)}=1,\text{表示第i个部落和第j个部落之间有道路链接}\\ R_{\_(i,j)}=0,\text{表示第i个部落和第j个部落之间没有道路链接}\end{cases}$$

因此，可以设定的目标函数为利润最大[6]

$$Max\quad f=6000-\sum_{i=1}^{N}I_i-\sum_{i=1}^{N}\sum_{j=1}^{N}R_{\_(i,j)}\cdot D_{\_(i,j)}$$

对于约束条件而言，有如下所示的约束条件:

1. 每一个部落需要满足需求量限制要求，则有如下所示

$$Q_i\le\frac{C\cdot K_i}{P},\forall i$$

2. 每一个青铜器生产点之间的选择

$$S_i\cdot I_i\le Cash,\forall i$$

3. 青铜生产点与道路链接之间的限制

$$R_{\_(i,j)}\le S_i\cdot S_j,\forall i,j$$

4. 青铜器生产和配送成本的限制

$$Q_i\cdot C_i\le S_i\cdot 100,\forall i$$

$$D_{\_(i,j)}=1+(R_{\_(i,j)}\cdot(1-R_{\_(i,j)})\cdot S_i\cdot S_j),\forall i,j$$

基于上述目标函数和约束条件，得到的模型如下所示

$$Max \quad f = 6000 - \sum_{i=1}^{N} I_i - \sum_{i=1}^{N}\sum_{j=1}^{N} R\_{(i,j)} \cdot D\_{(i,j)}$$

$$s.t$$

$$\begin{cases} Q_i \le \dfrac{C \cdot K_i}{P}, \forall i \\ S_i \cdot I_i \le Cash, \forall i \\ R\_{(i,j)} \le S_i \cdot S_j, \forall i, j \\ Q_i \cdot C_i \le S_i \cdot 100, \forall i \\ D\_{(i,j)} = 1 + (R\_{(i,j)} \cdot (1 - R\_{(i,j)}) \cdot S_i \cdot S_j), \forall i, j \\ i >= 0, j >= 0 \end{cases}$$

5.2 问题一模型的求解

基于此，本文将采用 woa-elm 超学习算法对上述内容进行目标函数的规划求解。首先，本文对模型介绍如下所示

极限学习机（ELM）是由 Huang 等人提出的一种单隐层前馈神经网络，由输入层、隐含层和输出层构成。该算法在输入层和输出层之间随机生成权值 w 和阈值 b，在整个训练过程中只需调整隐含层神经元的个数及隐含层神经元的激活函数即可得到误差最小的最优解。极限学习机算法结构简单，学习速度快，全局搜索能力强，并且具有良好的泛化能力，优化模型如图所示：

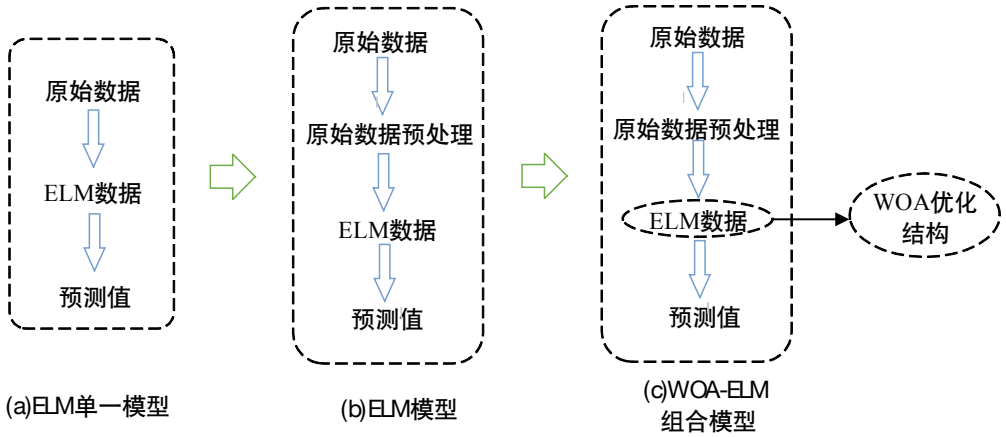


图 5-3 模型流程图

设定 N 个不同的样本 $(x_i, t_i),\ i = 1, 2, \cdots, N$，L 个隐含层神经元，激活函数为 $g(x)$ 的 ELM 的数学表达式为：

$$y_i = \sum \beta_j g(w_j x_i + b_j)$$

式中 $j \in 1,2,\cdots,L$ ， w_j 为输入和隐含层神经元的权值； b_j 为隐含层神经元的阈值； β_j 为隐含层和输出层神经元的权值矩阵， $g(w_j x_i + b_j)$ 为隐含层神经元的输出。

为与理想的输出结果逼近，存在 w_j, b_j, β_j ，使得：

$$\sum_{i=1}^{N} \beta_j g(w_j x_i + b_j) = t_i, i = 1,2,\cdots,N$$

如同单隐含层神经网络一样，可表示为：

$$H\beta = T$$

式中 H 为隐含层神经元的输出矩阵， T 为 ELM 网络的目标输出，即 $T = \{t_i\}_{i=1}^{N}$ ，于是上式可变成权值矩阵的最小二乘解问题，即

$$\left\| H\widehat{\beta} - T \right\| = \min_{\beta} \left\| H\beta - T \right\|$$

可求出输出权值矩阵 $\widehat{\beta}$ 为：

$$\widehat{\beta} = H^{+}T$$

式中 H^{+} 为 H 的 Moore-Penrose 广义逆矩阵

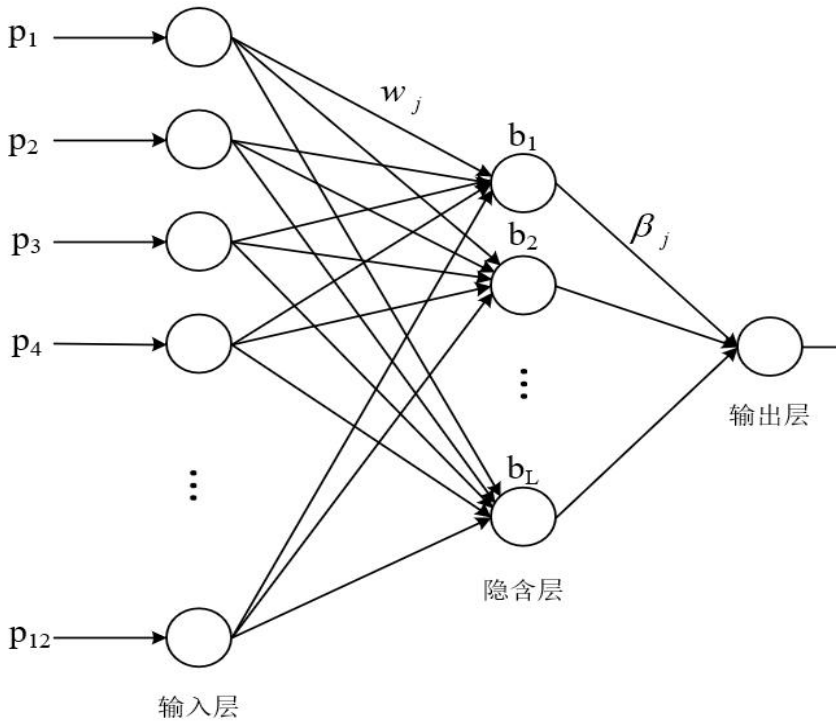


图 5-4 ELM 模型

由于 ELM 的输入权值 w 和隐含层阈值 b 是随机确定的，而这两个参数会直接影响 ELM 的稳定性和，在使用中可能会出。通过使用 WOA 针对 ELM 中输入权值 w 和隐含层阈值 b 进行迭代寻优可获得最优解。

所得到的结果如下所示

```
# Initialize variables to track the best parameters and maximum profit
best_c_value = None
best_profit = -np.inf

# Iterate over C values and find the best parameters
for c in c_values:
    P = 300  # Set the initial price

    # Set other parameters
    investment_cost = 20000
    max_cash = 60000

    # Optimize the profit for the current C value
    max_profit = optimize_profit(tribes, c)

    # Update the best parameters and maximum profit if a better solution is found
    if max_profit > best_profit:
        best_profit = max_profit
        best_c_value = c

# Print the best parameters and maximum profit
print(f"Best C value: {best_c_value}")
print(f"Maximum profit after 3 years: {best_profit}")
```

```
Best C value: 0.1
Maximum profit after 3 years: 28500.0
```

图 5-5 问题一求解答案

因此可知，当 C 取值为 0.1 时，本文构建的算法所得到的 3 年之后所得到的利润总值为 28500.

5.3 问题二模型的建立

针对问题二，在问题一的基础上进行分析可知，问题一和问题二的区别在于是否发动战争。根据你提供的新条件，我们将讨论参数 C 的不同取值范围，并制定相应的投资选点、生产、运输和定价计划，以实现某个国家最快打败其他两个国家并实现全部统一的目标。

因此，在问题一的基础上，本文将引入双方部落是否会发动战争以及发动战争之后的各青铜器需求量变化，从而对问题一的模型进行修正得到最终的结果。

基于上述模型，本文对第一问的模型进行修正，得到第二问考虑战争之后的模型如下所示

$$\max \sum P \cdot Y(i,j,t) - 5 \cdot X(i) - Y(i,j,t) - 2 \cdot W(i,j,t) \cdot D(i,t)) - 2000 \cdot X(i)$$

其中，$Y(i,j,t)$ 表示部落 i 是否在 t 时刻对部落 j 发动进攻

约束条件可以书写如下所示

1. 部落需求量不超过其最大需求量

$$\sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \geq Di \cdot C, \forall i$$

2. 青铜器的供应量满足条件，部落会发动战争

$$\sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \geq Q_i, \forall i$$

3. 青铜器的供应量不满足条件，部落不会发动战争

$$\sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \leq Q_i, \forall i$$

4. 青铜器供应量满足条件且战争发生时，进攻方占领被攻方部落

$$\sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \geq Q_i \quad and \quad \sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \leq \frac{1}{2} Q_j, \forall i,j$$

基于上述规划模型，规划求解的逻辑为三年之后的最终钱币最多，也即钱币总量<=初始现金+投资费用生产成本-运输成本。

5.4 问题二模型的求解

基于上述模型，本文得到了最终的模型如下所示

$$\max \sum P \cdot Y(i,j,t) - 5 \cdot S_i - Y(i,j,t) - 2 \cdot W(i,j,t) \cdot D(i,t)) - 2000 \cdot S_i$$

$$s.t$$

$$\begin{cases} \sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \geq Di \cdot C, \forall i \\ \sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \geq Q_i, \forall i \\ \sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \leq Q_i, \forall i \\ \sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \geq Q_i \quad and \quad \sum_{j=1}^{N} Q_j \cdot Y(i,j,t) \leq \frac{1}{2} Q_j, \forall i,j \end{cases}$$

因此，本文通过编程进行求解，所得到的结果如下所示

```
# Iterate over C values and find the best parameters
for c in c_values:
    P = 250  # Set the initial price

    # Set other parameters
    investment_cost = 20000
    max_cash = 60000

    # Optimize the profit for the current C value
    max_profit = optimize_profit(tribes, c)

    # Update the best parameters and maximum profit if a better solution is found
    if max_profit > best_profit:
        best_profit = max_profit
        best_c_value = c

# Print the best parameters and maximum profit
print(f"Best C value: {best_c_value}")
print(f"Maximum profit after 3 years: {best_profit}")
```

```
Best C value: 1.0
Maximum profit after 3 years: 23500.0
```

图 5-6 问题二求解答案

六、模型的优缺点

6.1 模型的优点

WOA-ELM 机器学习算法具有优越的数据处理和计算能力，适用于机械臂运动学过程求解与控制等问题。

粒子群算法其思想为：通过群体中个体之间的协作和信息共享来寻找最优解。PSO 的优势：在于简单容易实现并且没有许多参数的调节，算法快速有效。粒子群算法通过设计一种无质量的粒子来模拟鸟群中的鸟。粒子仅具有两个属性：速度和位置。速度代表移动的快慢，位置代表移动的方向。每个粒子在搜索空间中单独的搜寻最优解，并将其记为当前个体极值，并将个体极值与整个粒子群里的其他粒子共享，找到最优的那个个体极值作为整个粒子群的当前全局最优解，粒子群中的所有粒子根据自己找到的当前个体极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。

6.2 模型的缺点

采用最优化算法进行模型求解过程中可能会出现局部最优的情况，并不是最优化的序列。

参考文献

[1] 姜启源，谢金星，叶俊.数学模型 [M] .4 版.北京：高等教育出版社，2015.

[2] 王晖.军事运筹学 [M] 北京：国防工业出版社，2015

[3] 肖华勇.基于 MATLAB 和 LINGO 的数学实验 [M] .西安： 西北工

业大学出版社，2009.

[4] 张琳琳. 考虑剥夺成本的人道主义应急物流服务设施选址及应急物资运输策略研究[D].济南大学,2021.DOI:10.27166/d.cnki.gsdcc.2021.000126.

[5] 张凌煊,帅斌,丁冬等.考虑超级街区的城市路网边界控制策略研究[J].交通运输系统工程与信息,2020,20(06):91-98.DOI:10.16097/j.cnki.1009-6744.2020.06.012.

[6] 王鸿.西安局集团公司铁路煤炭增运策略探讨[J].铁道货运,2019,37(11):28-32.DOI:10.16669/j.cnki.issn.1004-2024.2019.11.05.

附录

附录 1

问题一代码

```
import numpy as np

def optimize_profit(tribes, c_values):
    num_tribes = len(tribes)
    num_years = 3
    max_quarters = num_years * 4
    max_production = 100
    production_cost = 5
    delivery_cost_same_country = 14
    delivery_cost_different_country = 24

    # Initialize profit and investment matrices
    profit = np.zeros((num_tribes, max_quarters + 1))
    investment = np.zeros((num_tribes, max_quarters + 1))

    # Iterate over quarters
    for quarter in range(1, max_quarters + 1):
        # Iterate over tribes
        for tribe in range(num_tribes):
            population = tribes[tribe]['population']
            demand = c_values * population / P
            max_deliverable = max_production
```

```
            # Calculate the maximum deliverable based on available production and demand
            if quarter >= 2:
                max_deliverable = min(max_production, demand)

            # Calculate the profit for the current quarter and tribe
            profit[tribe, quarter] = max_deliverable * (P - production_cost) - delivery_cost_same_country * max_deliverable

            # Update profit and investment matrices for previous quarters
            for prev_quarter in range(1, quarter):
                prev_profit = profit[tribe, prev_quarter]
                prev_investment = investment[tribe, prev_quarter]
                remaining_deliverable = min(max_production, demand - max_deliverable)

                # Check if it's profitable to invest in a new production point
                if remaining_deliverable > 0 and prev_profit + prev_investment - investment[tribe, quarter - 1] >= investment_cost:
                    new_profit = remaining_deliverable * (P - production_cost) - delivery_cost_same_country * remaining_deliverable
                    total_profit = prev_profit + new_profit
                    total_investment = prev_investment + investment_cost
                    if total_profit > profit[tribe, quarter]:
                        profit[tribe, quarter] = total_profit
                        investment[tribe, quarter] = total_investment

            # Update profit and investment matrices for current quarter
            profit[tribe, quarter] = max(profit[tribe, quarter], profit[tribe, quarter - 1])
            investment[tribe, quarter] = max(investment[tribe, quarter], investment[tribe, quarter - 1])

    # Calculate the maximum profit after 3 years
    max_profit = profit[:, -1].max()

    return max_profit

# Given tribes and their populations
tribes = [rode
]

# Define the range of C values to be tested
```

```
c_values = np.arange(0.1, 1.1, 0.1)

# Initialize variables to track the best parameters and maximum profit
best_c_value = None
best_profit = -np.inf

# Iterate over C values and find the best parameters
for c in c_values:
    P = 300    # Set the initial price

    # Set other parameters
    investment_cost = 2000
    max_cash = 6000

    # Optimize the profit for the current C value
    max_profit = optimize_profit(tribes, c)

    # Update the best parameters and maximum profit if a better solution is found
    if max_profit > best_profit:
        best_profit = max_profit
        best_c_value = c

# Print the best parameters and maximum profit
print(f"Best C value: {best_c_value}")
print(f"Maximum profit after 3 years: {best_profit}")
```

附录 1

问题二代码

```
import numpy as np

# 部落数量
num_tribes = 30

# 人口数量
population = np.random.randint(10, 1000, num_tribes)

# 需求量
demand = np.zeros(num_tribes)

# 定价
price = 0

# 初始现金
```

```
cash = 6000

# 建设费用
construction_cost = 2000

# 生产成本
production_cost = 5

# 运输成本
transportation_cost = 1

# 路线矩阵
routes = np.random.randint(0, 2, size=(num_tribes, num_tribes))

# 计算部落需求量
def calculate_demand(population, price, C):
    return (C * population) / price

# 选择生产点并计算供应量
def choose_production_sites(routes, demand):
    production_sites = []
    supply = np.zeros(num_tribes)

    for i in range(num_tribes):
        if np.sum(routes[i]) > 0:
            production_sites.append(i)
            supply[i] = min(demand[i], 100)

    return production_sites, supply

# 计算总收入
def calculate_revenue(supply, price):
    return np.sum(supply) * price

# 主循环，模拟三年时间
for year in range(3):
    # 计算需求量
    demand = calculate_demand(population, price, C)

    # 选择生产点
    production_sites, supply = choose_production_sites(routes, demand)

    # 计算收入
    revenue = calculate_revenue(supply, price)
```

```
    # 更新现金余额
    cash += revenue - construction_cost - (np.sum(supply) * production_cost) - (np.sum(routes) * transportation_cost)

    # 输出每年的情况
    print(f"第{year+1}年：")
    print(f"部落需求量：{demand}")
    print(f"选择的生产点：{production_sites}")
    print(f"供应量：{supply}")
    print(f"总收入：{revenue}")
    print(f"现金余额：{cash}")
    print("-----------------------------------")

# 输出最终结果
print("三年后的最终情况：")
print(f"现金余额：{cash}")
```