

实验报告：市区公共救护中心选址优化

一、问题

扩军与裁军问题敌对的两个国家都面临着两种选择：扩充军备或裁减军备。如果双方进行军备竞赛（扩军），都将为此付出3000亿美元的代价；如果双方都裁军，则可以省下这笔钱。但是倘若有一方裁军，另一方扩军，则扩军一方发动侵略战争，占领对方领土，从而可获益1万亿美元。裁军一方由于军事失败而又丧失国土则可以认为损失无限，试建立该问题的对策模型，并求该问题的纳什平衡解。

二、问题分析

在这个问题中，我们要建立一个博弈论模型来分析两个敌对国家在扩军与裁军问题上的对策选择。我们可以将这两个国家分别称为国家A和国家B，并构建它们的收益矩阵。为了便于分析，我们假设“无限损失”是一个非常大的负值。

收益矩阵的构建

让我们定义各方的行动及其对应的收益：

- 如果两国都选择扩军，双方的收益分别为 -3000亿美元（因为扩军成本是3000亿美元）。
- 如果两国都选择裁军，双方的收益分别为 0（因为没有成本也没有收益）。
- 如果一方选择扩军，另一方选择裁军，扩军的一方收益为 10000亿美元，裁军的一方损失非常大，可以认为是 $-\infty$ 。

纳什平衡解

纳什平衡是指在博弈中，任何一方在对手策略不变的情况下，无法通过单方面改变策略来提高自身收益的策略组合。

我们通过检查每一个策略组合来找到纳什平衡：

- (扩军, 扩军)**：在这种情况下，任何一方单方面改变策略为裁军将会导致无限损失。因此没有一方有动力改变策略。所以 (扩军, 扩军) 是一个纳什平衡。
- (裁军, 裁军)**：在这种情况下，任何一方单方面改变策略为扩军将会导致对方无限损失，而自己获得 10000亿美元的收益，因此 (裁军, 裁军) 不是一个纳什平衡。
- (扩军, 裁军) 和 (裁军, 扩军)**：在这两种情况下，选择裁军的一方将会遭受无限损失，因此这些组合不会稳定。

三、模型建立

1. 收益矩阵的构建

为了建立和求解这个博弈问题的数学模型，我们需要定义收益矩阵并使用博弈论的工具来求解纳什平衡。我们将使用Python中的库 `nashpy` 来帮助我们求解。

首先，我们定义两个玩家的收益矩阵。假设我们使用一个非常大的负数来表示无限损失，比如 `-1e9`。

收益矩阵

为了方便编程，我们使用一个非常大的负数来表示无限损失，比如 `-1e9`。

$$A \text{ 的收益矩阵} = \begin{bmatrix} -3000 & 10000 \\ -1e9 & 0 \end{bmatrix} \quad B \text{ 的收益矩阵} = \begin{bmatrix} -3000 & -1e9 \\ 10000 & 0 \end{bmatrix}$$

2. 变量定义

x, y 分别代表国家A和国家B的策略，且均为 2×1 的矩阵

$$y = [y_1, y_2]^T, x = [x_1, x_2]^T$$

3. 目标函数

$$\max (x^T A y - \nu_1 + x^T B y - \nu_2)$$

带入上面数据得到

$$\max (-3000x_1y_1 - (M - 10000)x_2y_1 + 1000y_2x_1 - Mx_1y_2)$$

4. 约束条件

$$\begin{aligned} Ay &\leq \nu_1 \mathbf{1}_m \\ B^T \mathbf{x} &\leq \nu_2 \mathbf{1}_n \\ \mathbf{x}^T \mathbf{1}_m &= 1 \\ \mathbf{y}^T \mathbf{1}_n &= 1 \\ \mathbf{x} &\geq 0 \\ \mathbf{y} &\geq 0 \end{aligned}$$

约束条件等价于

$$\begin{aligned} -3000y_1 - My_2 &\leq \nu_1 \\ 10000y_1 &\leq \nu_1 \\ -3000x_1 - Mx_2 &\leq \nu_2 \\ 10000x_1 &\leq \nu_2 \\ x_1 + x_2 &= 1 \\ y_1 + y_2 &= 1 \\ x_i &\geq 0, i = 1, 2 \\ y_j &\geq 0, j = 1, 2 \end{aligned}$$

四、模型求解

python代码

python自带nash问题的库，故直接调用即可实现

```
import nashpy as nash
import numpy as np

# 定义收益矩阵
# 使用 -1e9 表示无限大的负值
A_payoff_matrix = np.array([
    [-3000, 10000],
    [-1e9, 0]
```

```

])

B_payoff_matrix = np.array([
    [-3000, -1e9],
    [10000, 0]
])

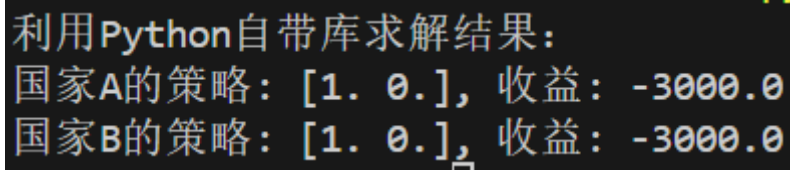
# 创建博弈
game = nash.Game(A_payoff_matrix, B_payoff_matrix)

# 求解纳什均衡
equilibria = list(game.support_enumeration())

# 打印结果
for eq in equilibria:
    A_strategy, B_strategy = eq
    A_value = np.sum(A_strategy[:, None] * B_strategy[None, :] *
A_payoff_matrix)
    B_value = np.sum(A_strategy[:, None] * B_strategy[None, :] *
B_payoff_matrix)
    print("利用Python自带库求解结果:")
    print(f"国家A的策略: {A_strategy}, 收益: {A_value}")
    print(f"国家B的策略: {B_strategy}, 收益: {B_value}")

```

得到结果如下图所示



```

利用Python自带库求解结果:
国家A的策略: [1. 0.], 收益: -3000.0
国家B的策略: [1. 0.], 收益: -3000.0

```

Lingo代码求解

```

MODEL:
sets:
str1/1..2/:x;
str2/1..2/:y;
rew(str1,str2):a,b;
endsets

data:
a = 2 5,
    0 4;
b = 2 0,
    5 4;
enddata

v1 = @sum(rew(i,j):a(i,j)*x(i)*y(j));
v2 = @sum(rew(i,j):b(i,j)*x(i)*y(j));
@free(v1);
@free(v2);
@for(str1(i):@sum(str2(j):a(i,j)*y(j))<=v1);
@for(str2(j):@sum(str1(i):b(i,j)*x(i))<=v2);
@sum(str1:x)=1;

```

```
@sum(str2:y)=1;  
END
```

求解结果如下

```
Feasible solution found.  
Infeasibilities: 0.000000  
Total solver iterations: 9  
Elapsed runtime seconds: 0.09
```

```
Model Class: QP
```

```
Total variables: 6  
Nonlinear variables: 4  
Integer variables: 0
```

```
Total constraints: 8  
Nonlinear constraints: 2
```

```
Total nonzeros: 24  
Nonlinear nonzeros: 6
```

Variable	Value
V1	2.000000
V2	2.000000
X(1)	1.000000
X(2)	0.000000
Y(1)	1.000000
Y(2)	0.000000

六、结论

综上所述，国家A与国家B选取的策略应当都为扩军，与之前分析一致