# Report

**181220076 周韧哲**

## 运行方式

命令行输入`python main.py`即可计算纳什均衡并写入`output`文件夹中。

## 实现方式

对于两人博弈，我使用的算法是Labeled Polytopes Algorithm，对于多人博弈计算所有PNE，我是直接计算每个Player的最优反应并判断是否为NE。

### 两人博弈

**算法思路：**

- 只需要求解半空间的极点即可：

$$
\begin{aligned}
P_1 &= \left\{ \mathbf{x} \in \mathbb{R}^m : \mathbf{x} \geq 0, \mathbf{B}^\top \mathbf{x} \leq \mathbf{1} \right\}, \\
P_2 &= \left\{ \mathbf{y} \in \mathbb{R}^n : \mathbf{y} \geq 0, A\mathbf{y} \leq \mathbf{1} \right\}.
\end{aligned}
$$

- 对于给定的极点$x, y$，令$L(x), L(y)$分别为使等号成立的不等式下标集合：

$$
\begin{aligned}
L(\mathbf{x}) &= \left\{ k \in S_1 : x_k = 0 \right\} \cup \left\{ j \in S_2 : (B^\top \mathbf{x})_l = 1 \right\}, \\
L(\mathbf{y}) &= \left\{ l \in S_2 : y_l = 0 \right\} \cup \left\{ i \in S_1 : (A\mathbf{y})_k = 1 \right\}.
\end{aligned}
$$

- $L(x), L(y)$称为$x, y$的label，若$L(x) \cup L(y) = S_1 \cup S_2$，则$(x, y)$称为fully label，且为NE。

**具体实现：**

具体计算过程如下：

1. 首先将$P_1$的约束$x \geq 0, B^T x \leq 1$转换为$\hat{B}x \leq b$，同理，$P_2$的约束转换为$\hat{A}y \leq a$。
2. 计算$P_1$和$P_2$的极点。为了计算半空间$P_1$：$\hat{B}x \leq b$的极点，先求解优化问题：

$$
\begin{aligned}
\max \quad & y \\
\text{s.t.} \quad & \hat{B}x + y\|\hat{B}_i\|_2 \leq b
\end{aligned}
$$

   其中$\|\hat{B}_i\|_2$代表$\hat{B}$的按行求2范数形成的列向量。使用scipy的线性规划求解器解出$x$，然后通过半空间交集求解器便可得到每个极点。（参考见这个[document]）。同理可以求得$P_2$的极点。
3. 对于$P_1$和$P_2$的极点$v_1, v_2$，计算其满足哪几个等号条件从而得到其label $L(v_1), L(v_2)$，如果$L(x) \cup L(y)$等于全label，则$(v_1, v_2)$就是一个NE。

计算两人博弈的PNE和MNE为`lp_alg()`和`get_pole()`函数，详细请见代码。

### 多人博弈

假设有$n$个Player，第$i$个Player的策略集合为$A_i$，payoff矩阵为$P_i$，具体计算过程如下：

1. 对于每个Player，假设为$i$，计算其他Player所有的可能反应，即其他所有Player的策略笛卡尔积：$\prod_{j=1, j\neq i}^{n} A_j$，对于每个反应，通过找$P_i$中相应的最大值来计算Player $i$ 的最优反应，并保存起来。

2. 找到所有Player中保存的最优反应，筛选出在每个Player的最优反应都出现了的反应，这些相同的最优反应即为PNE。

计算多人博弈的PNE为`build_pne()`函数，详细请见代码。

## 结果分析

- 对于两人博弈，在策略空间（$|A|$）小的时侯，能够很快解出所有NE；但在策略空间很大，比如payoff矩阵为$10 \times 10$时，P1和P2的极点可能有很多，需要较长时间才能匹配完所有label，解出所有NE。比如对于第13,14,15个文件，P1和P2的极点数量达到了三位数，匹配label花费了主要运行时间，总运行时间在2~5秒左右，而对于其他的文件运行时间不足0.01秒。
- 对于多人博弈，所有PNE都能很快求解出来。因为尽管payoff矩阵可能很大，但固定其他Player的反应后的当前Player最优反应数较少，所以匹配时间短。