纳什均衡的计算与复杂性（一）

2024-2025 学年春夏学期计算经济学讨论班 均衡的计算

金政羽

Clovers2333@gmail.com

浙江大学计算机科学与技术学院

2025 年 **2** 月 **21** 日

Contents

Basic Theorems

定理 (Nash Theorem)

Every finite game has at least one Nash equilibrium.

定义 (最佳响应 / Best Response)

For a player i, a strategy s_i is a best response to a strategy profile s_{-i} if $U_i(s_i, s_{-i}) \geq U_i(s_i', s_{-i})$ for all s_i'.

定理 (无差异原则)

令 σ^* 为一个混合策略纳什均衡，令 s_i 为参与人 i 的一个纯策略，如果 $\sigma^*(s_i) > 0$ 那么 s_i 是 σ^*_{-i} 的最佳响应.

并且，记 s_i, s_i' 均是纯策略，如果 $\sigma^*(s_i) > 0, \sigma^*(s_i') = 0$，那么 $U_i(s_i, \sigma^*_{-i}) \geq U_i(s_i', \sigma^*_{-i})$；如果 $\sigma^*(s_i) > 0, \sigma^*(s_i') > 0$，那么 $U_i(s_i, \sigma^*_{-i}) = U_i(s_i', \sigma^*_{-i})$.

Basic Theorems

定理 (无差异原则)

更进一步，对于一个混合策略 σ^*，如果对于任何参与人 i 的任何一个纯策略 s_{ij}，都有 $\sigma^*(s_{ij}) = 0$ 或者 s_{ij} 是 σ^*_{-i} 的最佳响应，那么 σ^* 是一个纳什均衡.

根据以上定理，我们称参与人 i 的纯策略 s_{ij} 是被纳什均衡 σ^* 支持的，如果 $\sigma^*(s_{ij}) > 0$.

如果我们在求解纳什均衡的时候，可以提前知道每个参与人的哪些纯策略是被支持的，那么我们将可以把问题简化成一个线性方程组问题，并且多项式计算.

因此，求解纳什问题的难点，主要在于如何确定哪些纯策略是被支持的. 基于以上结论，我们将引出 Lemke-Howson 算法，它不仅给出了混合策略纳什均衡的普遍解法，也通过构造算法来证明了双人博弈的情况下的纳什定理.

Bimatrix Game and Symmetric Game

定义 (Bimatrix Game)

When the players play mixed strategies with vectors x and y, the expected payoff of the row player is $x^\top Ay$ and of the column player: $x^\top By$.

定义 (Symmetric Game)

A game with a payoff of $U_i : A_1 \times A_2 \times \cdots \times A_n \to \mathbb{R}$ for player i, where A_i is player i's strategy set and $A_1 = A_2 = \cdots = A_N$ is considered symmetric if for any permutation π:

$$U_{\pi(i)}(a_1, \ldots, a_i, \ldots, a_N) = U_i(a_{\pi(1)}, \ldots, a_{\pi(i)}, \ldots, a_{\pi(N)}).$$

对于 Bimatrix Game，如果它是 Symmetric Game，那么我们有 $U_1(x,y) = U_2(y,x)$. 因此，我们可以简单令 $U_1(x,y) = x^\top Ay$, $U_2(y,x) = y^\top A^\top x$.

Reduction to Symmetric Games I

定理

There is a polynomial-time reduction from Bimatrix Game to Symmetric Game.

证明.

对于 $U_1(x,y) = x^\top Ry, U_2(y,x) = y^\top Cx$ 的 Bimatrix Game (其中 x 是一个 m 维向量，y 是一个 n 维向量)，我们可以构造矩阵:

$$A = \begin{bmatrix} 0 & R \\ C^\top & 0 \end{bmatrix}$$

Reduction to Symmetric Games II

证明（续）.

通过矩阵 A，我们可以构造一个 Symmetric Game：

$$U_1'(x',y') = x'^\top A y', U_2'(x',y') = y'^\top A^\top x'$$

其中 x', y' 都是一个 $m+n$ 维向量；令 x_1, y_1 为 m 维向量，x_2, y_2 为 n 维向量，$x' = (x_1, x_2), y' = (y_2, y_1)$.

所以我们有：

$$U_1'(x',y') = U_2'(y',x') = x_1^\top R y_1 + x_2^\top C^\top y_2$$

假设 x^*, y^* 是 Symmetric Game 的 Nash Equilibrium，$x^* = (x_1^*, x_2^*)$，$y^* = (y_2^*, y_1^*)$，那么我们有：(x_1^*, x_2^*) 是 (y_2^*, y_1^*) 的 Best Response，同理 (y_2^*, y_1^*) 是 (x_1^*, x_2^*) 的 Best Response.

Reduction to Symmetric Games III

证明（续）.

所以:

$$
\begin{aligned}
x_1^* &= \arg\max_{x_1} x_1^\top R y_1 + x_2^\top C^\top y_2 \\
&= \arg\max_{x_1} x_1^\top R y_1 \\
y_2^* &= \arg\max_{y_2} x_1^\top R y_1 + x_2^\top C^\top y_2 \\
&= \arg\max_{y_2} x_2^\top C^\top y_2
\end{aligned}
$$

因此，我们可以得到 x_1^*, y_2^* 是原 Bimatrix Game 的 Nash Equilibrium. □

接下来，我们都将基于双人 Symmetric Game 进行讨论.

Strategies and Inequalities

首先需要声明，对称游戏虽然双方的策略空间、收益函数都是一样的，但是这并不代表两个玩家在纳什均衡的时候所选择的混合策略一定是相同的.

我们接下来所讨论的 Lemke-Howson 算法，它寻找纳什均衡是基于"均衡时双方混合策略相同"的条件；这个算法"一定有解"的性质也将说明，对于对称游戏，一定存在双方的混合策略都相同的纳什均衡.

Strategies and Inequalities

定义 (纳什均衡点)

双人 Symmetric Game 中，双方的决策空间都是 $\mathbb{R}^n$. 如果在达到纳什均衡时，双方都选择相同混合策略 y，那么我们称 y 是这个对称游戏的一个纳什均衡点.

知道了纳什均衡点的定义，我们如何从这个定义出发在 $\mathbb{R}^n$ 的空间中找到均衡点呢?

Strategies and Inequalities

纳什均衡点

对于一个纳什均衡点 $y \in \mathbb{R}^n$，需要满足以下性质：

- $0 \leq y_i \leq 1$;
- $\sum_{i=1}^n y_i = 1$.
- (y, y) 满足纳什均衡条件，即：假设第一个玩家的收益矩阵是 A，那么有：$\forall x \in \mathbb{R}^n, x^\top Ay \leq y^\top Ay$.

接下来我们重点分析第二个和第三个性质.

对于第二个条件，事实上，我们完全不需要管这个等式，我们只需要知道 y 向量各个维度之间的比例即可，我们要做的只是求解出 y 向量通过等比例放缩后的另一个向量 z.

Strategies and Inequalities

纳什均衡点

对于一个纳什均衡点 $y \in \mathbb{R}^n$，需要满足以下性质：

- $0 \leq y_i \leq 1$;
- $\sum_{i=1}^n y_i = 1$.
- (y, y) 满足纳什均衡条件，即：假设第一个玩家的收益矩阵是 A，那么有：$\forall x \in \mathbb{R}^n, x^\top Ay \leq y^\top Ay$.

对于第三个性质，我们将套用之前的"无差异原则"：

无差异原则

s_i, s_i' 均是纯策略，若 $\sigma^*(s_i) > 0, \sigma^*(s_i') = 0$，则 $U_i(s_i, \sigma^*_{-i}) \geq U_i(s_i', \sigma^*_{-i})$；若 $\sigma^*(s_i) > 0, \sigma^*(s_i') > 0$，则 $U_i(s_i, \sigma^*_{-i}) = U_i(s_i', \sigma^*_{-i})$.

更进一步，对于一个混合策略 σ^*，如果对于任何参与人 i 的任何一个纯策略 s_{ij}，都有 $\sigma^*(s_{ij}) = 0$ 或者 s_{ij} 是 σ^*_{-i} 的最佳响应，那么 σ^* 是一个纳什均衡.

Strategies and Inequalities

无差异原则

s_i, s_i' 均是纯策略，若 $\sigma^*(s_i) > 0, \sigma^*(s_i') = 0$，则 $U_i(s_i, \sigma^*_{-i}) \geq U_i(s_i', \sigma^*_{-i})$；若 $\sigma^*(s_i) > 0, \sigma^*(s_i') > 0$，则 $U_i(s_i, \sigma^*_{-i}) = U_i(s_i', \sigma^*_{-i})$.

更进一步，对于一个混合策略 σ^*，如果对于任何参与人 i 的任何一个纯策略 s_{ij}，都有 $\sigma^*(s_{ij}) = 0$ 或者 s_{ij} 是 σ^*_{-i} 的最佳响应，那么 σ^* 是一个纳什均衡.

设纳什均衡点 $y = (y_1, y_2, \cdots, y_n)$，$value = y^\top A y$，我们有：

若 $y_i > 0$，那么 $e_i^\top A y = A_i y = value$，其中 A_i 是收益矩阵 A 的第 i 行；

若 $y_i = 0$，那么 $e_i^\top A y = A_i y \leq value$.

因此，y 是对称游戏的一个纳什均衡点当且仅当：

对于任意 i，$A_i y = value$ 和 $y_i = 0$ 二者必定有一个成立.

Inequalities and Polytope

记 $z = \dfrac{y}{value}$，我们有：

对于任意 i，$A_i z = 1$ 和 $z_i = 0$ 二者必定有一个成立.

因此，我们可以将 $2n$ 个等式 $A_i z = 1, z_i = 0$ 视为 $2n$ 个超平面，这些超平面在 n 维空间中围成的图形就是我们想要的多胞体（polytope）.

接下来我们的 Lemke-Howson 算法将在这个多胞体上进行.

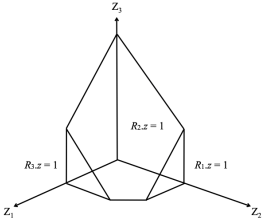
我们可以称这个多胞型是非退化的，若没有任何一个超平面是其他超平面的线性组合；事实上，我们可以通过非常微小的扰动使得任意的多胞型非退化.

Inequalities and Polytope

假设我们对称博弈的矩阵为:

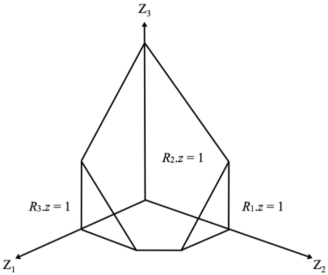
$$R = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 2 & 2 \\ 3 & 0 & 0 \end{pmatrix}, C = R^\top$$

那么 $2n$ 个不等式围成的多胞体如下:



这些超平面围成的空间可以表示为 $A_i z \leq 1, z_i \geq 0, i = 1, 2, \cdots, n.$

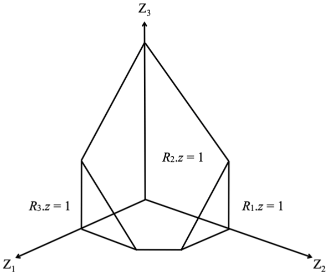
Inequalities and Polytope



观察这个多胞型，我们可以发现如下性质：

- 对于每个顶点，它都是 n 个超平面的交点，即 n 个不等式同时取等号.
- 每个顶点都有 n 个相邻的顶点，即 n 个不等式取等号的情况下，改变一个不等式的取等号，就可以到达另一个顶点.

Inequalities and Polytope

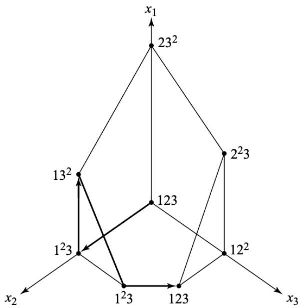


在 $2n$ 个等式中，每个顶点一定都是满足其中的 n 个，我们将 $z_i = 0$ 和 $A_i z = 1$ 合称为第 i 组：

若顶点被第 i 组中的一个等式的超平面穿过，我们称这个顶点被 i 表示一次；如果被两个超平面穿过，我们称这个顶点被 i 表示两次.

Inequalities and Polytope

在多胞型的顶点进行编号，如下图所示：



我们要找的纳什均衡点，就是除了原点以外，编号为 $1, 2, \cdots, n$ 的顶点.

Lemke-Howson Algorithm

Lemke-Howson Algorithm 算法的基本思路是：首先构造一个无向图，在相邻的两个点之间进行连边；其次构造一个有向路径，从原点出发，走到我们最终想要的纳什均衡点.

任意选择一个特殊的组，记其为 k. 在所有的顶点中，我们只关注三类顶点：

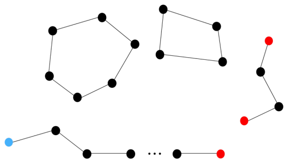
1. 原点；
2. 除了原点以外，编号为 $1, 2, \cdots, n$ 的顶点，也就是我们要找的纳什均衡点；
3. 在 n 个组中，仅仅有一个组没有被表示，或者说仅仅有一个组被表示了两次.

Lemke-Howson Algorithm

对于第 1 类和第 2 类顶点，我们仅连出一条边：松弛第 k 组的等式，然后在直线上进行移动，直到找到相邻的顶点;

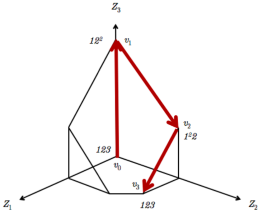
对于第 3 类顶点，我们设其重复两次的组为 l，我们分别松弛 l 组中的两个等式，将会连出两条边.

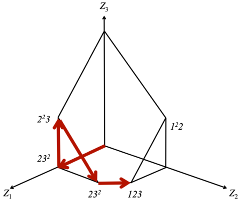
连边后的效果如下:



容易发现，从原点出发，我们一定可以找到一条路径，走到我们想要的纳什均衡点.

Lemke-Howson Algorithm





选取不同的特殊组 k，我们有可能会游走到不同的纳什均衡点，所以对于任意一个 Bimatrix Game, 我们利用 Lemke-Howson 算法，最多可以找到 $n+m$ 个纳什均衡点.

Is Nash Equilibrium NP-Complete?

叙述完求解一般纳什均衡的 Lemke-Howson 算法后，我们将分析这个算法的时间复杂度.

我们试图将纳什均衡问题纳入已知的复杂类，首先我们很容易发现纳什均衡问题是一个 NP 问题，那么更进一步，它是不是 NP-Complete 问题呢？

Is Nash Equilibrium NP-Complete?

叙述完求解一般纳什均衡的 Lemke-Howson 算法后，我们将分析这个算法的时间复杂度.

我们试图将纳什均衡问题纳入已知的复杂类，首先我们很容易发现纳什均衡问题是一个 NP 问题，那么更进一步，它是不是 NP-Complete 问题呢？

比较纳什均衡问题和 SAT 我们发现一个很大的区别：纳什均衡一定有解！

Is Nash Equilibrium NP-Complete?

叙述完求解一般纳什均衡的 Lemke-Howson 算法后，我们将分析这个算法的时间复杂度.

我们试图将纳什均衡问题纳入已知的复杂类，首先我们很容易发现纳什均衡问题是一个 NP 问题，那么更进一步，它是不是 NP-Complete 问题呢？

比较纳什均衡问题和 SAT 我们发现一个很大的区别：纳什均衡一定有解！

那么如果纳什均衡是 NP-Complete 问题，会发生什么呢？

Is Nash Equilibrium NP-Complete?

定理 (Megiddo and Papadimitriou)

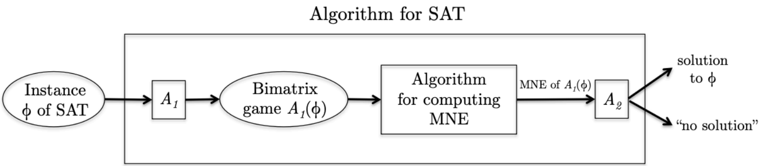
如果纳什均衡是 *NP-Complete* 问题，那么 *NP=co-NP*.

证明.

若纳什均衡是 NP-Complete 问题，那么我们存在一个从 SAT 问题到纳什均衡问题的多项式时间规约，其中，这个规约算法包含两个函数：

- A polynomial-time algorithm A_1 that maps every SAT formula ϕ to a Bimatrix Game $A_1(\phi)$;
- A polynomial-time algorithm A_2 that maps every Nash Equilibrium (x, y) of a game $A_1(\phi)$ to a satisfying assignment $A_2(x, y)$ of ϕ,if one exists,and to the string "no"otherwise.

Is Nash Equilibrium NP-Complete?

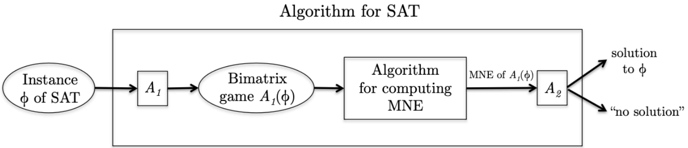


证明（续）.

因为"可以规约"，所以对于一个可以满足的公式，我们只要找到规约问题的一个纳什均衡点，就一定可以找到一个满足的解；

所以对于 A_2 函数，只要公式可以满足，那么任意纳什均衡点 (x, y) 都可以通过 A_2 函数映射到满足的解.

Is Nash Equilibrium NP-Complete?



证明（续）.

考虑一个不可能被满足的公式 ϕ，我们首先将其转化为一个 Bimatrix Game $A_1(\phi)$，此时如果我们有 $A_1(\phi)$ 的一个纳什均衡 (x, y)，我们可以通过调用 A_2，发现该函数将会返回 No.

由此我们得到了一个可以多项式判定"SAT 问题是否一定不可解"的算法，因此 SAT 也是 $co-NP$ 的.

而 SAT 问题是 NP-Complete 问题，所以 $NP \subseteq co-NP$，这两个问题又是对偶的，所以 $NP = co-NP$. □

TFNP Class

定义 (TFNP)

如果一个问题，对于任何输入，都至少有一个解，并且解可以在多项式时间内进行验证，那么它就属于 TFNP 复杂类.

显然 TFNP 是 NP 问题的一个子集，Nash Equilibrium 问题是一个 TFNP 问题，既然纳什均衡不能是范围更广的 NP-Complete 问题，那么它能不能是范围稍窄一点的 TFNP-Complete 问题呢？

TFNP Class

定义 (TFNP)

如果一个问题，对于任何输入，都至少有一个解，并且解可以在多项式时间内进行验证，那么它就属于 TFNP 复杂类.

显然 TFNP 是 NP 问题的一个子集，Nash Equilibrium 问题是一个 TFNP 问题，既然纳什均衡不能是范围更广的 NP-Complete 问题，那么它能不能是范围稍窄一点的 TFNP-Complete 问题呢？

不幸的是，我们目前没有找到一个 TFNP-Complete 问题.

我们称 NP 还有 P 这些为句法类问题（syntactic），即其成员可以通过具体的计算模型（确定性 / 不确定性图灵机）定义，其 Complete 问题也可以定义在这样的计算模型上；

但是对于 TFNP 我们称其为语义类问题（semantic），例如求解纳什均衡、因式分解，前者的存在性依赖拓扑学的知识，后者的存在性依赖数论的知识，我们无法建立起一个统一的模型将这些知识组合起来.

PPAD Class

因此，我们希望进一步深挖 TFNP 问题，给纳什均衡的求解问题一个更精确的复杂类.

PPAD (for polynomial parity argument, directed version)

- 有向图：图的顶点表示问题的状态，每条有向边表示从一个状态到另一个状态的转移;
- 每个顶点的入度和出度最多为 1;
- 图中存在一个已知的源顶点（入度为 0），作为路径跟踪的起点；目标是找到另一个源顶点或汇顶点（出度为 0），作为问题的解.

Nash Equilibrium belongs to PPAD

根据 Lemke-Howson 算法，我们已经将一般的纳什均衡转化成了一个在多胞体上游走的问题，我们可以看看这个问题是否符合 PPAD 的定义.

Nash Equilibrium and PPAD

- 有向图：每个顶点表示一个纳什均衡点，每条边表示从一个纳什均衡点到另一个纳什均衡点的转移；
- 在选取特殊组 k 后，每个顶点的入度和出度最多为 1；
- 我们已经知道了原点，并且目标的纳什均衡必然是另一个源顶点或汇顶点.

所以本质上，Nash Equilibrium 问题和 PPAD 的定义性问题是等价的，因此 Nash Equilibrium 就是 PPAD-Complete 的.

Brouwer Fixed Point and PPAD

另一个比较经典的 PPAD 问题是 Brouwer Fixed Point Theorem，它的定义如下：

Brouwer Fixed Point Theorem

Let D be a closed, bounded, and convex subset of $\mathbb{R}^n$, and let $f : D \to D$ be a continuous function. Then there exists a point $x \in D$ such that $f(x) = x$.

（比较抽象）我们可以将函数 f 看成一个有向图，由 x 射向 $f(x)$，容易发现，每个顶点的入度和出度最多均为 1，我们可以从任意点开始，直到找到没有出度的点——即我们要找的不动点.

定理 (Hirsch et al., 1989)

Any algorithm for finding Brouwer fixpoints that treats the function as a black box must be exponential.

Are PPAD-Complete Problems Hard?

我们首先大致估计一下 PPAD 问题的难度：它一定比 NP 问题简单，因为多了一个“一定有解”的条件；同时也肯定比 P 困难.

我们先前论证了 PPAD 是 NPC 问题的后果，那如果 PPAD 是 P 问题呢?

如果 PPAD 问题可以多项式解决，它将颠覆目前很多已有的认知——纳什均衡本质上就是线性规划；对于每个函数，我们可以通过分析它的特殊性质来更快速地计算出它的不动点.

看上去确实很“科幻”，但毕竟我们都没有证明 $P \neq NP$，如果 $P = NP$，那么自然会有 $P = PPAD = NP$.

局部搜索问题：最大割

接下来，我们先把混合策略的纳什均衡放在一边，讨论一下更特殊的情况——纯策略纳什均衡，以及求解纯策略纳什均衡的算法及其复杂性.

首先我们介绍一个非常熟悉的问题——最大割问题.

局部搜索问题：最大割

学习局部搜索的一个经典例子就是最大割问题．最大割问题的输入是一张无向图 $G=(V,E)$，其中每条边 $e\in E$ 的权重是 w_e $(w_e\geq 0)$．该问题的可行解是一个割 (X,X')，即图的顶点集 V 划分为两部分．该问题的目标是将割边的权重之和最大化，割边就是那些一个顶点在集合 X 中，另一个顶点在集合 X' 中的边．如果 $P\neq NP$，则该问题不存在多项式时间的算法．

局部搜索是一种很自然的启发式算法，它适用于很多 NP 困难问题，包括最大割问题．局部搜索的算法非常简单．

最大割问题的局部搜索算法

初始化任意一个割 (X,X')
while 存在更优的局部移动 **do**
    执行任意一个这样的局部移动

局部搜索问题：最大割

这时局部移动的意思是，将一个点 v 从割的一个侧移动到另一个侧．例如，如果将节点 v 从集合 X 移动到 X'，则目标函数的增量为：

$$\sum_{u\in X,(u,v)\in E} w_{uv} - \sum_{u\in X',(u,v)\in E} w_{uv}$$

如果以上公式中的差是正数，那么这样的一个局部移动就是更优的．如果找不到更优的局部移动，局部搜索算法就结束了，结果时产生的结果就是局部最优．局部最优不一定是全局最优.

局部搜索问题：最大割

我们可以将局部搜索算法可视化，展示为图中的有向图 H 上的一个行走．

对于最大割问题，它的输入是一个图 G，那么相对应的，在可视化的有向图 H 中，每条有向边表示一个更优的局部移动．这些局部移动间不应该有环，所以图 H 是一个有向无环图．图 H 中没有出边的节点（吸收点）表示局部最优．局部搜索算法在图 H 中不断地沿着出边行走，直到进入一个吸收点为止．

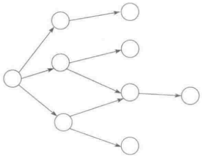


图: 局部搜索算法的可视化

PLS：抽象局部搜索问题

我们希望定义一种新的复杂类，它将更严格地框定最大割局部搜索的复杂度，并且也将用来框定我们即将引出的纯策略纳什均衡问题.

PLS 的定义

PLS 是一个局部搜索问题，并且满足以下条件：

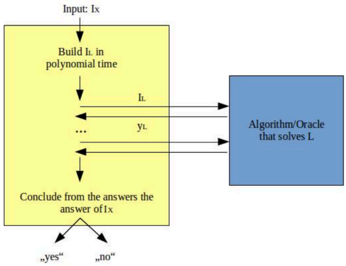
- 第一个多项式时间的算法：输入是一个局部搜索问题的情境，输出是任意一个可行解.
- 第二个多项式时间的算法：输入是一个局部搜索问题的情境和它的一个可行解，输出是这个解的目标函数数值.
- 第三个多项式时间的算法：输入是一个局部搜索问题的情境和它的一个可行解，这个算法要么报告当前的解就是局部最优解，要么产生一个具有更好目标函数数值的解.

PLS：抽象局部搜索问题

定理

If a PLS Problem is NP-hard, then NP=co-NP.

之前已经证明过了 TFNP 的形式，而 $PLS \subseteq TFNP$，所以显然.



A First PLS-Complete Problem: Circuit / Flip

定义 (Circuit / Flip)

An instance I of Circuit/Flip is an acyclic boolean circuit with $x_1, \ldots, x_n$ inputs and $y_1, \ldots, y_m$ outputs. It can be imagined for example as an actual circuit or a $m \times 1$ vector of boolean expressions. A solution s of I is a certain assignment of $x_1, \ldots, x_n$. The cost of a solution $x_1, \ldots, x_n$ is the output $y_1, \ldots, y_m$ read as an integer number, so:

$$c(x_1, \ldots, x_n) = \sum_{i=1}^{m} 2^{(i-1)}(y_i)$$

很显然，Max - Circuit / Flip 与 Min - Circuit / Flip 是等价的.

纯纳什均衡是 PLS 问题

在纯策略纳什均衡时，我们定义”neighborhood” 为一个玩家改变一个策略的情况，即从一个纯策略到另一个纯策略.

假如当前所有人的策略集合是 s，定义势能函数 $f(s)$ 为所有玩家的收益之和，那么每次一个玩家发现策略的偏移，一定是函数 $f(s)$ 有一个势能更大的邻居；当局部搜索达到局部最优解后，玩家也没有了偏移的动机——达到了纳什均衡.

A First PLS-Complete Problem: Circuit / Flip

Example 3.1 Max-circuit/Flip

$$\begin{pmatrix} x_1 \vee x_2 \\ (x_1 \vee x_2) \wedge \neg x_3 \end{pmatrix}$$

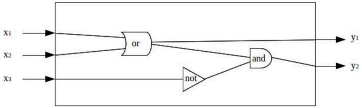


Figure 3: Circuit I

Table 3: Solutions and costs

One solution s:	$x_1 = 1,\ x_2 = 1,\ x_3 = 1$
Output of solution s:	$y_1 = 1,\ y_2 = 0$
Cost of solution s:	as binary: 01, as decimal: 1
Neighbors of solution s:	$x_1 = 1,\ x_2 = 1,\ x_3 = 0$ with output $y_1 = 1,\ y_2 = 1$ and cost in decimal 3
	$x_1 = 1,\ x_2 = 0,\ x_3 = 1$ with output $y_1 = 1,\ y_2 = 0$ and cost in decimal 1
	$x_1 = 0,\ x_2 = 1,\ x_3 = 1$ with output $y_1 = 1,\ y_2 = 0$ and cost in decimal 1
Local optimum:	$x_1 = 1,\ x_2 = 1,\ x_3 = 0$ with output $y_1 = 1,\ y_2 = 1$ and cost in decimal 3

A First PLS-Complete Problem: Circuit / Flip

Example 3.1 Max-circuit/Flip

$$\begin{pmatrix} x_1 \lor x_2 \\ (x_1 \lor x_2) \land \neg x_3 \end{pmatrix}$$

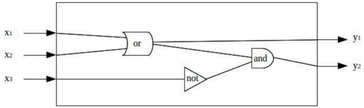


Figure 3: Circuit I

Table 3: Solutions and costs

One solution s:	$x_1 = 1,\ x_2 = 1,\ x_3 = 1$
Output of solution s:	$y_1 = 1,\ y_2 = 0$
Cost of solution s:	as binary: 01, as decimal: 1
Neighbors of solution s:	$x_1 = 1,\ x_2 = 1,\ x_3 = 0$ with output $y_1 = 1,\ y_2 = 1$ and cost in decimal 3
	$x_1 = 1,\ x_2 = 0,\ x_3 = 1$ with output $y_1 = 1,\ y_2 = 0$ and cost in decimal 1
	$x_1 = 0,\ x_2 = 1,\ x_3 = 1$ with output $y_1 = 1,\ y_2 = 0$ and cost in decimal 1
Local optimum:	$x_1 = 1,\ x_2 = 1,\ x_3 = 0$ with output $y_1 = 1,\ y_2 = 1$ and cost in decimal 3

To be Continued...

- PLS-Complete 问题的规约;
- PPAD-Complete 问题的规约;
- 近似纳什均衡的引入及相关定理;
- ……