Chapter 2

Computation of Equilibria

In this chapter, we will examine the complexity of computing Nash equilibria in mixed extensions of two-player games. One reason to restrict ourselves to games with two players is that in these games, a Nash equilibrium with rational strategies is guaranteed to exist provided that the input data is rational. This is not the case for three-player games, see Exercise 1.29.

It will be convenient to assume that the two-player game is given by two matrices $A, B \in \mathbb{R}^{m \times n}$, where m is the number of strategies of player 1 and n is the number of strategies of player 2. In the following, we will refer to such games as bimatrix games. In this chapter, it will be convenient, to assume that the

$$S_1 = \{1, \ldots, m\} \qquad \text{and} \qquad S_2 = \{m+1, \ldots, m+n\}.$$

It is also convenient, to denote mixed strategies of the first players with $\mathbf{x}$ and mixed strategies of the second player with $\mathbf{y}$. Recall that

$$\Delta(S_1) = \left\{\mathbf{x} \in \mathbb{R}^m_{\geq 0} : \mathbf{x}^\top \mathbf{1} = 1\right\} \qquad \text{and} \qquad \Delta(S_2) = \left\{\mathbf{y} \in \mathbb{R}^n_{\geq 0} : \mathbf{y}^\top \mathbf{1} = 1\right\},$$

where $\mathbf{1}$ denotes an all-ones vector in appropriate dimension. Recall from Definition 1.9 that the players' utilities are defined as $u_i(x, y) = \sum_{k \in S_1} \sum_{l \in S_2} u_i(k, l) x_k y_l$, i.e.,

$$u_1(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top A \mathbf{y} \qquad \text{and} \qquad u_2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top B \mathbf{y}.$$

Then, Corollary 1.11 can be reformulated as follows.

Corollary 2.1 (Best-Response Condition (BRC)). Let $(\mathbf{x}, \mathbf{y})$ be a Nash equilibrium with $\operatorname{supp}(\mathbf{x}) = T_1$ and $\operatorname{supp}(\mathbf{y}) = T_2$ for some $T_1 \subseteq S_1$ and $T_2 \subseteq S_2$. Then, there are $u, v \in \mathbb{R}$ such that

$$\begin{aligned} (A\mathbf{y})_k &= u \quad \text{for all } k \in T_1, & (A\mathbf{y})_k &\leq u \quad \text{for all } k \in S_1 \setminus T_1,\\ (\mathbf{x}^\top B)_l &= v \quad \text{for all } l \in T_2, & (\mathbf{x}^\top B)_l &\leq v \quad \text{for all } l \in S_2 \setminus T_2, \end{aligned}$$

```
foreach $(T_1, T_2)$ with $T_1 \subseteq S_1$ and $T_2 \subseteq S_2$ do
    Solve the linear feasibility problem

        $x \geq 0$,        $(Ay)_k \geq (Ay)_{k'}$          for all $k \in T_1, k' \in S_1$,
        $y \geq 0$,        $(x^\top B)_l \geq (x^\top B)_{l'}$  for all $l \in T_2,\ l' \in S_2$,
        $x^\top 1 = 1$,    $x_k = 0$                       for all $k \in S_1 \setminus T_1$,
        $y^\top 1 = 1$,    $y_l = 0$                       for all $l \in S_2 \setminus T_2$

    if the system has a solution $(x^*, y^*)$ then return it
end
```

Algorithm 2.1: Computing mixed Nash equilibria by support enumeration.

2.1 Support Enumeration

Corollary 1.11 gives rise to a simple algorithm computing a Nash equilibrium of a mixed extension of a two-player finite game. Simply try all possible support sets T_1 and T_2 and solve a system of linear equations that check whether the conditions of Corollary 1.11 can be satisfied simultaneously by stochastic vectors x and y, see Algorithm 2.1.

Example 2.2. We illustrate one iteration of Algorithm 2.1 with the game in Figure 2.1 as input. For $T_1 = \{1, 2\} \subseteq S_1$ and $T_2 = \{4, 5\} \subseteq S_2$, we obtain that player 2 must be indifferent between its two columns 1 and 2, i.e.,

$$3x_1 + 2x_2 = 2x_1 + 6x_2 \quad \Leftrightarrow \quad x_1 = 4x_2,$$

which together with $x_1 + x_2 = 1$ yields that $x_1 = 4/5$ and $x_2 = 1/5$. Similarly, player 1 must be indifferent between its two rows 1 and 2, i.e.,

$$3y_4 + 3y_5 = 2y_4 + 5y_2 \quad \Leftrightarrow \quad y_4 = 2y_5,$$

which together with $y_4 + y_5 = 1$ yields $y_4 = 2/3$ and $y_5 = 1/3$. Thus, (x, y) with $x = (4/5, 1/5, 0)$ and $y = (2/3, 1/3)$ is a Nash equilibrium.

As a corollary of this procedure we obtain that finite two-player games always have a rational mixed Nash equilibrium.

Corollary 2.3. Any finite two player game with rational utility matrices has a rational mixed Nash equilibrium (x, y).

Proof. Any two-player game has a mixed Nash equilibrium (x, y) by Theorem 1.14. This implies that there are $T_1 \subseteq S_1$ and $T_2 \subseteq S_2$ such that the linear feasibility system in Algorithm 2.1 has a solution. The corresponding linear feasibility problem is clearly bounded so that the set of feasible points is a non-empty polytope in

$$\begin{array}{c|cc|} & 4 & 5 \\ \hline 1 & (3,3) & (3,2) \\ 2 & (2,2) & (5,6) \\ 3 & (0,3) & (6,1) \\ \hline \end{array}$$

(a) Representation as bi-matrix

$$A = \begin{pmatrix} 3 & 3 \\ 2 & 5 \\ 0 & 6 \end{pmatrix} . \quad \text{and} \quad B = \begin{pmatrix} 3 & 2 \\ 2 & 6 \\ 3 & 1 \end{pmatrix} .$$

(b) Representation as two matrices A, B

Figure 2.1: Bi-matrix game (Example 2.2).

some Euclidean space. Being non-empty the polytope has a vertex which can be described as the intersection of finitely many halfspaces that define valid equations or inequalities of the system. The vertex can then be computed by solving a system of linear equations. Using that the utility matrices are rational, these linear equations only have rational coefficients. When solving this system by Cramer's rule, it is apparent that every coefficient of the solution has only rational coefficients. □

2.2 Labeled Polytopes

Not every combination of subsets of pure strategies may occur as the support of a Nash equilibrium and it makes sense to search for these supports in a more systematic fashion. In the following, we restrict our attention to games that are non-degenerate in the following sense.

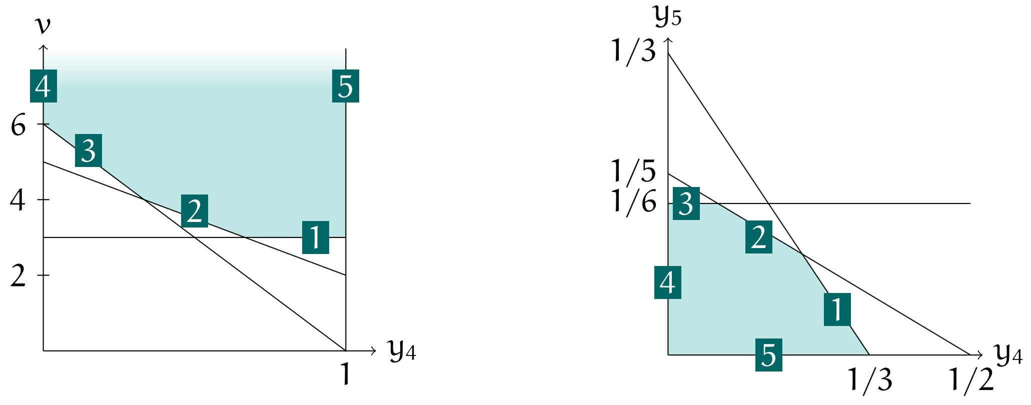
Definition 2.4. A bi-matrix game is non-degenerate if for every $(x, y) \in \Delta(S_1) \times \Delta(S_2)$
- $|\text{supp}(x)| \geq |\text{supp}(y)|$, if y is a best response to x,
- $|\text{supp}(y)| \geq |\text{supp}(x)|$, if x is a best response to y.

Assumption 2.5. The bimatrix game (A, B) is non-degenerate.

There are several possibilities to deal with degeneracies, such as perturbations of the matrices, and lexicographic arguments, but we are not going to elaborate on this issue here.

Theorem 2.6. In a Nash equilibrium (x, y) of a non-degenerate bi-matrix game, $|\text{supp}(x) = |\text{supp}(y)|$.

Proof. Follows from non-degeneracy and the fact that in a Nash equilibrium the two players' strategies are best responses to each other. □



(a) Best reply polyhedron **(b)** Normalized best reply polyhedron

Figure 2.2: Best response polyhedron $\bar{P}_2$ and normalized best reply polyhedron P_2 of player 2 for the game in Example 2.2. **(a)** The bounding halfspaces are shown with their respective label. Halfspaces with labels 1 to 3 enforce that pure strategies 1 to 3 of player 1 achieve no utility larger than v; halfspaces with labels 4 and 5 correspond to the restriction that y_4 and y_5 are non-negative. **(a)**

We further assume that the matrices A and $B^\top$ have nor zero column.

| Assumption 2.7. A and B have no zero-column.

This assumption can be made without loss of generality since adding a large constant to every entry of both matrices does not change the equilibria of the game. We now capture the best responses of each player by means of a best response polyhedron. Let

$$\bar{P}_1 := \left\{ (x, v) \in \mathbb{R}^m \times \mathbb{R} \;\middle|\; x \geq 0, B^\top x \leq u\mathbf{1}, x^\top \mathbf{1} = 1 \right\},$$
$$\bar{P}_2 := \left\{ (y, u) \in \mathbb{R}^n \times \mathbb{R} \;\middle|\; y \geq 0, Ay \leq v\mathbf{1}, y^\top \mathbf{1} = 1 \right\}.$$

The polyhedron $\bar{P}_1$ contains tuples (x, u) of mixed strategies x of player 1 together with a scalar u such that u is an upper bound on the expected utility of player 2. Analogously, the polyhedron $\bar{P}_2$ contains tuples (y, v) of mixed strategies y of player 2 together with a scalar v that is an upper bound on the utility of player 1 when player 2 plays y. We are interested in the extreme points of said polyhedra, because they satisfy some of the constraints with equality and, as a consequence, have several special properties.
Let us first consider $\bar{P}_1$, the best response polyhedron of player 1. If one of the constraints $x_k \geq 0$ is satisfied with equality, then $x_k = 0$ and the pure strategy $k \in S_1$ is not played in x. If one of the constraints $(B^\top x)_l \leq u$ is satisfied with equality, then the pure strategy $l \in S_2$ is a best response of player 2 to x. The polytope lives in $\mathbb{R}^{m+1}$ and has one equality constraint, so it is an m-dimensional polyhedron.

This implies that in any extreme point, m of the inequality constraints have to be satisfied with equality. An extreme point (x, u) thus corresponds to a situation where $|\text{supp}(x)| = k$ for some $k \leq m$ and at least k pure strategies of player 2 are a best response to x. By non-degeneracy, in fact exactly k pure strategies of player 2 are a best response. Anologously, at every extreme point (y, v) of $\bar{P}_2$, we have $|\text{supp}(y)| = k$ for some $k \leq n$ and exactly k pure strategies of player 1 are best responses to y.
By Theorem 2.6 and Corollary 2.1, every equilibrium of a non-degenerate game can be described as a pair of extreme points of $\bar{P}_1$ and $\bar{P}_2$. In order to find an equilibrium it is sufficient to find a pair of extreme points so that the strategies in the support of the extreme point of the one polytope are exactly those that are a best reply to the strategy of the other player in the other polytope.
To simplify the search for such extreme points, we normalize the polyhedra. First note that $u, v > 0$ since $A, B > 0$. Let

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

The vectors in the normalized best response polyhedra are no longer stochastic, but there is a direct correspondence between the extreme points of $\bar{P}_i$ and P_i, except for the zero vector. For each extreme point (x, u) of $\bar{P}_i$, x/u is an extreme point of P_i. For each extreme point x of P_i, apart from the zero vector, $(\frac{x}{x^\top \mathbf{1}}, \frac{1}{x^\top \mathbf{1}})$ is an extreme point of $\bar{P}_i$. In the following, for a non-zero vector x, we write $\bar{x}$ as a shorthand for $\frac{x}{x^\top \mathbf{1}}$.
We proceed to give a characterization of those extreme points of P_1 and P_2 that together consititute an equilibrium of the underlying game. For given extreme points x and y, let $L(x)$ and $L(y)$ be the index sets of those constraints that hold with equality, i.e.,

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

We refer to the sets $L(x)$ and $L(y)$ as the labels of x and y and call a pair $(x, y) \in P_1 \times P_2$ fully labeled if $L(x) \cup L(y) = S_1 \cup S_2$.
We first need the following useful lemma.

Lemma 2.8. In a non-degenerate game, for a pair of extreme points (x, y) we have $|L(x)| = m$ and $|L(y)| = n$.

Proof. Since the game is non-degenerate, at most $\text{supp}(\bar{x})$ pure strategies can be a best response to $\bar{x}$, and at most $\text{supp}(\bar{y})$ pure strategies can be a best response to $\bar{y}$, so

$$L(x) \leq |S_1 \setminus \mathrm{supp}(x)| + \mathrm{supp}(x) = m,$$
$$L(y) \leq |S_2 \setminus \mathrm{supp}(y)| + \mathrm{supp}(y) = n.$$

On the other hand, since x and y are extreme points of a full dimensional polytope, we have that $|L(x)| \geq m$ and $|L(y)| \geq n$, so that $|L(x)| = m$ and $|L(y)| = n$. $\square$

Theorem 2.9. A pair of extreme points $(x, y) \in P_1 \times P_2 \setminus \{(0,0)\}$ is fully labeled if and only if the vector $(\bar{x}, \bar{y})$ is a Nash equilibrium.

Proof. Let $(x, y) \in P_1 \times P_2 \setminus \{(0,0)\}$ be fully labeled. Let

$$T_1 = \mathrm{supp}(x) \qquad \text{and} \qquad T_2 = \mathrm{supp}(y).$$

We check that the conditions of Corollary 2.1 are satisfied. For all $k \in T_1$, x does not have label k as $x_k > 0$. From there, y must have label k, i.e., $(Ay)_k = 1$ and, thus, $(A\bar{y})_k = \frac{1}{y^\top 1}$ while $(A\bar{y})_k \leq \frac{1}{y^\top 1}$ for all $k' \in S_1$. This implies that k is a best reply to $\bar{y}$.
Further, for $k \notin T_1$, x does have label k. By Lemma 2.8, a label cannot appear twice in a fully labeled pair implying that y does not have label k, i.e, k is not a best reply to $\bar{y}$.
Conversely, assume that $(\bar{x}, \bar{y})$ is a Nash equilibrium. Then, by Corollary 2.1,

$$S_1 \setminus \mathrm{supp}(x) \cup \mathrm{supp}(y) \subseteq L(x),$$
$$S_2 \setminus \mathrm{supp}(y) \cup \mathrm{supp}(x) \subseteq L(y).$$

and, thus, $L(x) \cup L(y) = S_1 \cup S_2$. $\square$

2.3 The Lemke-Howson-Algorithm

The relationship between completely labeled pairs of extreme points and equilibria can be used to obtain a combinatorial algorithm for finding a Nash equilibrium of a bimatrix games. The idea is to start from the artificial pair $(0, 0)$ and pivot alternatingly in P_1 and P_2 until a completely labeled pair is found. Formally, let V_i be the sets of extreme points of P_i and let E_i by the set of edges between adjacent extreme points, i.e., extreme points that have all but two labels in common, i.e.,

$$E_1 = \{(x, x') \in V_1 \times V_1 : |L(x) \cap L(x')| = m - 1\},$$
$$E_2 = \{(y, y') \in V_2 \times V_2 : |L(y) \cap L(y')| = n - 1\}.$$

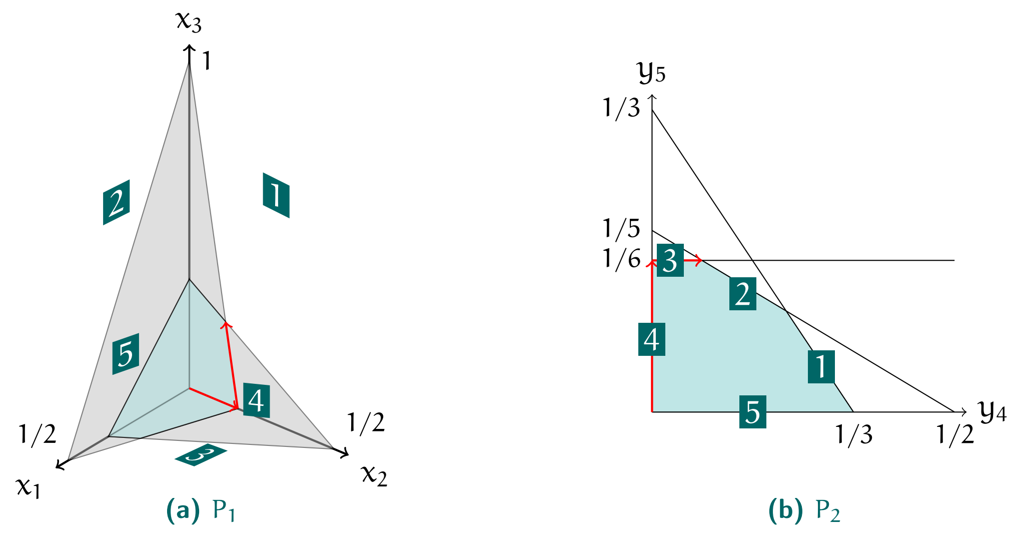


Figure 2.3: Run of The Lemke-Howson-Algorithm for the game of Example 2.2. The dropped label is 2. The final pair of extreme points (x, y) is $x = (0, 1/8, 1/4)^\top$, $y = (1/12, 1/6)^\top$.

Further, let

$$\begin{aligned}
V &= V_1 \times V_2, \\
E &= \{\big((x, y), (x', y)\big) \in V \times V : (x, x') \in E_1\} \\
&\quad \cup \{\big((x, y), (x, y')\big) \in V \times V : (y, y') \in E_2\}.
\end{aligned}$$

The key observation is that if we restrict our attention to extreme points that are almost fully labeled, with the possible exception of a particular label i, then there is always a unique way in which we can proceed. To this end, for $i \in S_1 \cup S_2$, let

$$\begin{aligned}
V_i &= \{(x, y) \in V : L(x) \cup L(y) \supseteq S_1 \cup S_2 \setminus \{i\}\}, \\
E_i &= E \cap (V_i \times V_i).
\end{aligned}$$

Theorem 2.10. Let $i \in S_1 \cup S_2$. Then, V_i contains $(0, 0)$ as well as every pair $(x, y) \in V$ such that $(\bar{x}, \bar{y})$ is a Nash equilibrium.
If the underlying game is non-degenerate, then $(0, 0)$ and the elements of V_i corresponding to an equilibrium have degree one in the graph (V_i, E_i), and all other nodes in V_i have degree two.

Proof. Both $(0, 0)$ and all pairs corresponding to equilibria are full labeled, so the first part is obvious.
For the second property, consider $(x, y) \in V_i$ and let $\bar{x}$ and $\bar{y}$ be the corresponding normalized strategies. Recall that by Lemma 2.8, $L(x) = m$ and $L(y) = n$. If $(x, y) = (0, 0)$ or $(\bar{x}, \bar{y})$ is an equilibrium, then (x, y) is fully labeled and $L(x) \cap L(y) = \emptyset$.
The neighbors of (x, y) are those elements in V_i that replace i with some other label, i.e., those where some other constraint holds with equality instead of the

```
Let $i \in S_1$ be arbitrary
$x \leftarrow 0$, $y \leftarrow 0$, $i'' \leftarrow i$
while true do
    In $P_1$, remove label $i''$ from $x$; let $x'$ be the new vertex and $i'$ the label added
    $x \leftarrow x'$
    if $i' = i$ then return $(\bar{x}, \bar{y})$
    In $P_2$, remove label $i'$ from $y$; let $y'$ be the new vertex and $i''$ the label added
    $y \leftarrow y'$
    if $i'' = i$ then return $(\bar{x}, \bar{y})$
end
```
Algorithm 2.2: The Lemke-Howson Algorithm

one corresponding to i. Since only x or y has label i, but not both, we may only replace it from one of them. Dropping the label i, we obtain a new label, and by non-degeneracy, this label is unique.
Otherwise, if $L(x) \cap L(y) = \{j\}$ for some duplicate label $j \in S_1 \cup S_2$. Then, the neighbors of (x, y) are obtained by replacing j with another label. This replacement can be done either in P_1 or P_2, and for each of them there is exactly one neighbor by the same reasoning as before. $\square$

We have shown that the graph (V_i, E_i) for each $i \in S_i \cup S_2$ consists of paths and cycles that are pairwise disjoint, and the ends of the path correspond to the pair $(0, 0)$ and to the equilibria of the underlying games. The algorithm of Lemke and Howson starts with the artificial pair of extreme points $(0, 0)$ and follows the path until it reaches the pair of extreme points at the other end, which is guarenteed to corrspond to an equilibrium. Moving from one vertex (x, y) of the graph to another corresponds to dropping a label from either x or y and picking up another label. In the first round, the dropped label is the deliberately chosen label i. In subsequent rounds it is the duplicate label that has been picked up in the previous round. Eventually, the missing label i will be picked up, and a fully labeled pair is reached, see Algorithm 2.2.
We have shown the following result.

Theorem 2.11. The Lemke-Howson-Algorithm outputs a Nash equilibrium.

As a byproduct of our analysis, we obtain that in a non-degenerate game, the number of Nash equilibria is odd.

Corollary 2.12. Every non-degenerate bimatrix game has an odd number of Nash equilibria.

The algorithm of Lemke and Howson is guaranteed to find a Nash equilibrium after a finite number of steps. However, there are examples that show that the number of needed pivots can be exponential.

Theorem 2.13 (Savani and von Stengel, 2006). There is a family of instances on which the Lemke-Howson-Algorithm takes exponential time.

2.4 Complimentary Pivoting

Adding slack variables $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$ turns the best response conditions into

$$B^\top \mathbf{x} + \mathbf{b} = \mathbf{1} \qquad \text{and} \qquad A\mathbf{y} + \mathbf{a} = \mathbf{1}, \tag{2.1}$$

where all vectors $\mathbf{x}, \mathbf{y}, \mathbf{a}, \mathbf{b} \geq 0$. The pair $(\mathbf{x}, \mathbf{y})$ is then completely labeled if and only if the orthogonality conditions

$$\mathbf{x}^\top \mathbf{a} = 0 \qquad \text{and} \qquad \mathbf{y}^\top \mathbf{b} = 0$$

are satisfied.
A basic solution to (2.1) is given by m linearly independent columns of the system $B^\top \mathbf{x} + \mathbf{b} = 1$ and n linearly independent columns of the system $A\mathbf{y} + \mathbf{a} = 1$. The non-basic variables are set to zero, so that the basic variables are uniquely determined. A basic solution that fulfills the non-negativity constraints a defines a pair of vertices of the normalized best reply polyhedra P_1 and P_2. The labels of a basic solution are given by the corresponding indices of the non-basic columns.
Pivoting describes the process of changing a basis where a single non-basic variable enters and a single basic variable leaves the set of basic variables while preserving feasibility. The label of the entering non-basic variable is dropped and the label of the leaving basic variable is picked up.
Reconsider the game from Example 2.2. The system $B^\top \mathbf{x} + \mathbf{b} = \mathbf{1}$ is equal to

$$\begin{aligned} 3x_1 + 2x_2 + 3x_3 + \ \ b_4 \qquad &= 1 \\ 2x_1 + 6x_2 + 1x_3 \qquad + \ \ b_5 &= 1. \end{aligned} \tag{2.2}$$

Here, b_4 and b_5 are the basic variables and the basic solution is $b_4 = b_5 = 1$.
Assume that label 2 is dropped. In this case, this corresponds to adding variable x_2 to the basis. Increasing x_2 while maintaining all equalities implies that $b_4 = 1 - 2x_2$ and $b_5 = 1 - 6x_2$, we can increase b_2 without turning one of the current basic variables negative as long as $b_2 \leq 1/6$. Formally, the term $1/6$ is the minimum ratio between the right hand side and the entry in the pivot column among those that have a positive entry in the latter. The row where the minimum is attained in the pivot row. As the minimum ratio is obtained for b_5, this is the basic variable that will leave the basis.
Algebraically, a pivoting step applies row operations to the system such that the column corresponding to the variable that enters the basis is a unit vector. This

can be done by dividing the pivot row by the pivot element and subtracting the pivot row from all other rows, i.e.,

$$\begin{aligned} \tfrac{7}{3}x_1 \quad\quad + \tfrac{8}{3}x_3 + b_4 - \tfrac{1}{3}b_5 &= \tfrac{2}{3} \\ \tfrac{1}{3}x_1 + x_2 + \tfrac{1}{6}x_3 \quad\quad + \tfrac{1}{6}b_5 &= \tfrac{1}{6}. \end{aligned} \tag{2.3}$$

Now, variable b_5 has left the basis and the label 5 appears in both systems. We thus turn to the system $Ay + a = 1$. The intial system is

$$\begin{aligned} 3y_4 + 3y_5 + a_1 \quad\quad &= 1 \\ 2y_4 + 5y_5 \quad + a_2 \quad &= 1 \\ 0y_4 + 6y_5 \quad\quad + a_3 &= 1. \end{aligned} \tag{2.4}$$

In order to remove the duplicate label 5, we have to make y_5 enter the basis. After a pivot step as above, we obtain

$$\begin{aligned} 3y_4 + \quad + a_1 \quad - \tfrac{1}{2}a_3 &= \tfrac{1}{2} \\ 2y_4 + \quad\quad + a_2 - \tfrac{5}{6}a_3 &= \tfrac{1}{6} \\ 0y_4 + y_5 \quad\quad + \tfrac{1}{6}a_3 &= \tfrac{1}{6}. \end{aligned} \tag{2.5}$$

We now return to system (2.3) to let x_3 enter the basis.

$$\begin{aligned} \tfrac{7}{8}x_1 \quad + x_3 + \tfrac{3}{8}b_4 - \tfrac{1}{8}b_5 &= \tfrac{1}{4} \\ \tfrac{3}{16}x_1 + x_2 \quad - \tfrac{1}{16}b_4 + \tfrac{3}{16}b_5 &= \tfrac{1}{8}. \end{aligned} \tag{2.6}$$

Return to system (2.4), we let y_4 enter the basis and obtain

$$\begin{aligned} a_1 - \tfrac{3}{2}a_2 + \tfrac{3}{4}a_3 &= \tfrac{1}{4} \\ y_4 \quad\quad + \tfrac{1}{2}a_2 - \tfrac{5}{12}a_3 &= \tfrac{1}{12} \\ y_5 \quad\quad + \tfrac{1}{6}a_3 &= \tfrac{1}{6}. \end{aligned} \tag{2.7}$$

Thus, we obtain the missing label 2 and obtain a fully labeled pair. We obtain the strategies $x_3 = 1/4$, $x_2 = 1/8$, and $y_4 = 1/12$, $y_5 = 1/6$. After renormalization this yields $\bar{x} = (0, 1/3, 2/3)$ and $\bar{y} = (1/3, 2/3)$.

2.5 Notes on Complexity

We have seen that a mixed equilibrium for two-player zero-sum game can be computed efficiently by linear programming techniques. In contrast, the Lemke-Howson-Algorithm may take an exponential number of steps until an equilibrium is reached. It is natural to ask whether there is a polynomial algorithm to compute a Nash equilibrium of a bimatrix game, i.e., whether there is a polynomial time algorithm for the following computational problem.

NASH
INPUT: Bimatrix game (A, B).
OUTPUT: Nash equilibrium (x, y) of (A, B).

To date, no polynomial algorithm is for this problem is known, and it is open whether such an algorithm exists. Thus, we would like to state that algorithm is computationally hard for a certain complexity class. However, computational complexity is traditionally only studied for decision problems. In a decision problem, only the answers “yes” and “no” are possible. As an example, consider the SATISFIABILITY problem below.
SAT
INPUT: Boolean formula ϕ in conjunctive normal form.
OUTPUT: “yes”, if ϕ has a solution, “no” otherwise.

Also for the SAT problem no polynomial algorithm is known. On the other hand, the famous Cook-Levin theorem states that the problem is NP-complete, i.e., it is as hard as any other problem in NP. Roughly speaking, the class NP consists of decision problems with the property for each instances I of the problem for which the correct answer is “yes”, there is a certificate x such that one can verify in polynomial time that I is indeed a “yes”-instance. We first want to give a formal definition of NP. There are many equivalent formulations of NP. We here use a definition that relies on relations.

Definition 2.14 (Polynomial-Time Recognizable Relation, Polynomially Balanced Relation). Let Σ be a finite. The relation $R \subseteq \Sigma^* \times \Sigma^*$ is called polynomial-time recognizable, if there is a polynomial p and an algorithm A that decides whether $(I, x) \in R$ in time $p(|I|)$. The relation is polynomially balanced, if $(I, x) \in R$ implies $|x| \leq p(|I|)$ for some polynomial p.

Definition 2.15 (NP, P). The class NP is the class of problems Π_R of the kind
INPUT: $I \in \Sigma^*$
OUTPUT:“yes”, if there is $x \in \Sigma^*$ s.t. $(I, x) \in R$, “no”, otherwise,
where $R \subseteq \Sigma^* \times \Sigma^*$ is a polynomial-time recognizable and polynomially balanced relation. The class P is the subclass of problems in NP that can be solved in polynomial time.

The Cook-Levin-Theorem makes a statement about the difficulty of said decision problem, stating that the satisfiability problem is in fact as difficult as any other problem in class NP. Note that this is a highly non-trivial theorem as it makes a statement about any decision problem in the rich class of problems NP.

Theorem 2.16 (Cook, 1971). The satisfiability problem is NP-complete, i.e., any problem in NP can be reduced to the satisfiability problem.

The problem Nash that we consider of in this chapter is of a different nature, since it is not a search problem, i.e., it is not even contained in NP. However, many natural decision problems concerning Nash equilibria in a bimatrix game are NP-complete.

Theorem 2.17 (Gilboa and Zemel, 1989; Conitzer and Sandholm, 2008). Given the mixed extension of a symmetric two-player game G, the following problems are NP-complete to decide:
- Does G have at least two Nash equilibria?
- Given $k \in S_1$, does G have a Nash equilibrium x with $k \in \operatorname{supp}(x_1)$?
- Given $k \in S_1$, does G have a Nash equilibrium x with $k \notin \operatorname{supp}(x_1)$?

Proof. We reduce from Sat. Let ϕ be a Boolean formula in conjunctive normal form, V its set of variables, $L = \{v : v \in V\} \cup \{-v : v \in V\}$ its set of literals and $C \subseteq 2^L$ the set of its clauses. Let $n = |V|$. Consider the symmetric two-player game $G = (\{1,2\}, \mathbf{S}, \mathbf{u})$ with $S = S_1 = S_2 = L \cup V \cup C \cup \{f\}$ and the following utilities:

$$\begin{aligned}
u_1(v,l) = u_2(l,v) &= \begin{cases} n & \text{if } l \notin \{v,-v\} \\ 0 & \text{else} \end{cases} && \text{for all } v \in V, l \in L \\
u_1(v,s) = u_2(s,v) &= n-4 && \text{for all } v \in V, s \in S \setminus (L \cup \{f\}) \\
u_1(l,l') = u_2(l',l) &= \begin{cases} n-1 & \text{if } l \neq -l' \\ n-4 & \text{else} \end{cases} && \text{for all } l, l' \in L \\
u_1(l,s) = u_2(s,l) &= n-4 && \text{for all } l \in L, s \in S \setminus (L \cup \{f\}) \\
u_1(c,l) = u_2(l,c) &= \begin{cases} n & \text{if } l \notin c \\ 0 & \text{else} \end{cases} && \text{for all } c \in C, l \in L \\
u_1(c,s) = u_2(s,c) &= n-4 && \text{for all } c \in C, s \in S \setminus (L \cup \{f\}) \\
u_1(s,f) = u_2(f,s) &= 0 && \text{for all } s \in S \setminus \{f\} \\
u_1(f,f) = u_2(f,f) &= \epsilon && \\
u_1(f,s) = u_2(s,f) &= n-1 && \text{for all } s \in S \setminus \{f\}.
\end{aligned}$$

See Figure 2.4 for an example of this construction.
We claim that the so-defined game has the following properties:
- (f,f) is a Nash equilibrium.
- If $(l_1,\dots,l_n)$ is a satisfying assignment of ϕ, then (x,y) with $x_{l_i} = y_{l_i} = 1/n$ for all $i = 1,\dots,n$ is a Nash equilibrium.
- There are no other Nash equilibria.

	x_1	x_2	$+x_1$	$-x_1$	$+x_2$	$-x_2$	$x_1 \vee -x_2$	$-x_1 \vee x_2$	f
x_1	$n-4$	$n-4$	0	0	n	n	$n-4$	$n-4$	0
x_2	$n-4$	$n-4$	n	n	0	0	$n-4$	$n-4$	0
$+x_1$	$n-4$	$n-4$	$n-1$	$n-4$	$n-1$	$n-1$	$n-4$	$n-4$	0
$-x_1$	$n-4$	$n-4$	$n-4$	$n-1$	$n-1$	$n-1$	$n-4$	$n-4$	0
$+x_2$	$n-4$	$n-4$	$n-1$	$n-1$	$n-1$	$n-4$	$n-4$	$n-4$	0
$-x_2$	$n-4$	$n-4$	$n-1$	$n-1$	$n-4$	$n-1$	$n-4$	$n-4$	0
$x_1 \vee -x_2$	$n-4$	$n-4$	0	n	n	0	$n-4$	$n-4$	0
$-x_1 \vee x_2$	$n-4$	$n-4$	n	0	0	n	$n-4$	$n-4$	0
f	$n-1$	$n-1$	$n-1$	$n-1$	$n-1$	$n-1$	$n-1$	$n-1$	ϵ

Figure 2.4: Example of a game constructed in the proof of Theorem 2.17 for the Boolean formula $\phi = (x_1 \vee -x_2) \wedge (-x_1 \vee x_2)$.

The first claim is obvious, so we start showing the second claim. If $(l_1, \dots, l_n)$ satisfies ϕ and player 2 plays $y_{l_i} = 1/n$ for all $i = 1, \dots, n$, then each strategy l_i of player 1 gives $n-1$ since the literal assignment is valid. On the other hand, playing a negation of a literal that occurs in a solution gives player 1 a utility

$$\frac{1}{n}(n-4) + \frac{n-1}{n}(n-1) < n-1.$$

Playing some variable v gives utility

$$\frac{1}{n}0 + \frac{n-1}{n}n = n-1.$$

Playing some clause c gives utility at most

$$\frac{1}{n}0 + \frac{n-1}{n}n = n-1,$$

where we use that at least one of the literals played by player 2 occurs in the clause c. Thus, playing the literals of the solution with equal probability is a best-reply of player 1. By symmetry, this strategy vector is a Nash equilibrium.
We proceed to prove that there are no other Nash equilibria. The proof proceeds in different steps, iteratively narrowing the possible support of a Nash equilibrium. First, we note that if one of the players always plays f, then clearly f is the unique best reply of the other player as well.

Step 1: $\mathrm{supp}(x), \mathrm{supp}(y) \subseteq L \cup \{f\}$. Suppose both players play f with probability less than 1. Consider the expected social welfare $u_1 + u_2$ conditioned on the event that no player plays f. There is no outcome with welfare large than $2n-2$, and any outcome in which one of the players plays a variable or a clause has social welfare at most $n-4+n < 2n-2$. Thus, if one of the players plays a variable or a clause with strictly positive probability then, there is a player who receives less than $n-1$ conditioned on the fact that none of the players plays f. This player, however, could deviate profitable to f. From there, no variable or clause is played with positive probability in a Nash equilibrium.
Step 2: $\mathrm{supp}(x), \mathrm{supp}(y) \subseteq L$. If both players only play strategies in $L \cup \{f\}$ and $\{f\}$ is played with positive probability by player 2, then player 1 can deviate playing only f. Thus, we may assume that both players only play strategies in L.
Step 3: $x_l + x_{-l} = y_l + y_{-l} = 1/n$ for all $l \in L$. Suppose that for $l \in L$, the probability that player 1 plays l and $-l$ together is less than $1/n$. Then, the expected utility of the player 2 when playing the corresponding variable strategy is strictly larger than

$$\frac{1}{n}0 + \frac{n-1}{n}n = n-1,$$

and thus the original profile is not at equilibrium. Thus, for any player, the probability of playing l or $-l$ is exactly $1/n$.
Step 4: $x_l = y_l = 1/n$ or $x_{-l} = y_{-l} = 1/n$ for all $l \in L$. If there is a literal where one player puts probability on the positive literal, and the other player on the negative one, both have utility less than $n-1$ and would be better of switching to f.
In conclusion, the only candidates for Nash equilbria are the strategy profile where both players play the literals of a valid assignment of the variables. If there is a clause that is not satisfied by this assignment, then this gives utility n and the players could profit from a deviation. □

2.5.1 Search Problems and Decision Problems

The computational problem NASH to compute a Nash equilibrium of a bimatrix game is different from decision problems in two different ways. First, it is a search problem, i.e., a problem for which we do not expect to get a simple "yes" or "no" as an answer. Second, the decision problem for Nash equilibria is trivial, since by Nash's theorem a Nash equilibrium is guaranteed to exist. Let us formally define the classes FNP and FP which are the search problem analogues of the well-known classes NP and P.

Definition 2.18 (FNP, FP). The class NP is the class of problems Π_R of the kind
INPUT: $I \in \Sigma^*$
OUTPUT: $x \in \Sigma^*$, such that $x \in \Sigma^*$, if it exists, "no", otherwise,
where $R \subseteq \Sigma^* \times \Sigma^*$ is a polynomial-time recognizable and polynomially balanced relation. The class FP is the subclass of problems in FNP that can be solved in polynomial time.

The prototypical example of a problem in FNP is the problem FSAT. The only difference between the decision problem SAT and its function version FSAT is, that for a "yes"-instance I of SAT it is sufficient to return "yes" while for FSAT we require as output $x \in \Sigma^*$ such that $(I, x) \in R$.
FSAT
INPUT: Boolean formula ϕ in conjunctive normal form.
OUTPUT: Satisfying assignment, if it exists, "no", otherwise.

2.5.2 Reductions

There is a natural notion of reduction between search problems in FNP that is similar to the polynomial-time reductions between decision problems in NP.

Definition 2.19 (Polynomial-Time Reductions Among Search Problems). Given two search problems problems Π_R and Π_S, we say that Π_R reduces to function problem Π_S, if there are polynomial-time computable functions f and g such that $(I, g(x)) \in R \Leftrightarrow (f(I), x) \in S$.

A search problem Π_R is FNP-complete, if $\Pi_R \in$ FNP and every $\Pi_S \in$ FNP reduces to Π_R. The following theorem can be proven by following the arguments of the proof of the Cook-Levin theorem (Theorem 2.16).

Theorem 2.20. FSAT is FNP-complete.

Search problems, like FSAT are not easier than decision problems since a polynomial time algorithm for FSAT gives rise to a polynomial time algorithm for SAT. On the other hand, FSAT reduces to SAT since a satisfying assignment can be figured out by running a linear number of SAT-algorithms. This is a special property of SAT which is called self reducibility.

Theorem 2.21. SAT is self-reducible.

Using that SAT is self-reducible and by the Cook-Levin theorem one can even show that every NP-complete problem is self-reducible.

Theorem 2.22. Every NP-complete problem is self-reducible.

This strong relationship between decision problems and search problems allows to prove the following theorem.

Theorem 2.23. $\mathsf{P} = \mathsf{NP} \Leftrightarrow \mathsf{FP} = \mathsf{FNP}$.

Proof.
"$\Leftarrow$": Assuming $\mathsf{FP} = \mathsf{FNP}$, we know that FSAT can be solved in polynomial time, i.e., a solution to every Boolean formula ϕ in conjunctive normal form can be in computed in polynomial time. This implies that also SAT can be solved in polynomial time. Using that SAT is NP-complete this implies $\mathsf{P} = \mathsf{NP}$.
"$\Rightarrow$": Assuming $\mathsf{P} = \mathsf{NP}$, SAT can be solved in polynomial time. Since SAT is self-reducible also FSAT can be solved in polynomial time. Using that FSAT is FNP-complete, this implies $\mathsf{FP} = \mathsf{FNP}$. □

There are problems in NP that are (probably) not self-reducible, e.g., the following.

Theorem 2.24 (Khuller and Vazirani (1991)). Planar graph 4-coloring is not self-reducible, unless $\mathsf{NP} = \mathsf{P}$.

2.5.3 Total search problems

Many interesting problems are guaranteed to have a solution. example are the problem to 4-colour a planar graph. Also the problem NASH falls into this category. Since their natural decision version

Definition 2.25 (TFNP). TFNP is the subclass of problems in FNP such that the underlying relation R is total, i.e, for all $I \in \Sigma^*$, there is $x \in \Sigma^*$.

The following theorem gives an alternative representation of TFNP.

Theorem 2.26 (Megiddo and Papadimitriou (1991)). $\mathsf{TFNP} = \mathsf{F}(\mathsf{NP} \cap \mathsf{coNP})$, where $\mathsf{F}(\mathsf{NP} \cap \mathsf{coNP})$ is the class of search problems that takes as input an instance of a problem in $\mathsf{NP} \cap \mathsf{coNP}$ and computes a "yes"-certificate for all "yes"-instances and a "no"-certificate for all "no"-instances.

Proof. For a problem $\Pi \in \mathsf{NP}$ there is a relation $R_{\text{yes}} \subseteq \Sigma^* \times \Sigma^*$ such for all "yes"-instances there is $x \in \Sigma^*$ with $(I, x) \in R_{\text{yes}}$. Similarly, for a problem $\Pi \in \mathsf{coNP}$ there is a relation $R_{\text{no}} \subseteq \Sigma^* \times \Sigma^*$ such for all "no"-instances there is $x \in \Sigma^*$ with $(I, x) \in R_{\text{no}}$.
Thus, the relation $R = R_{\text{yes}} \cup R_{\text{no}}$ is clearly total, and thus, $\mathsf{F}(\mathsf{NP} \cap \mathsf{coNP}) \subseteq \mathsf{TFNP}$. On the other hand, we may turn any problem $\Pi_R \in \mathsf{TFNP}$ with relation $R \subseteq \Sigma^* \times \Sigma^*$

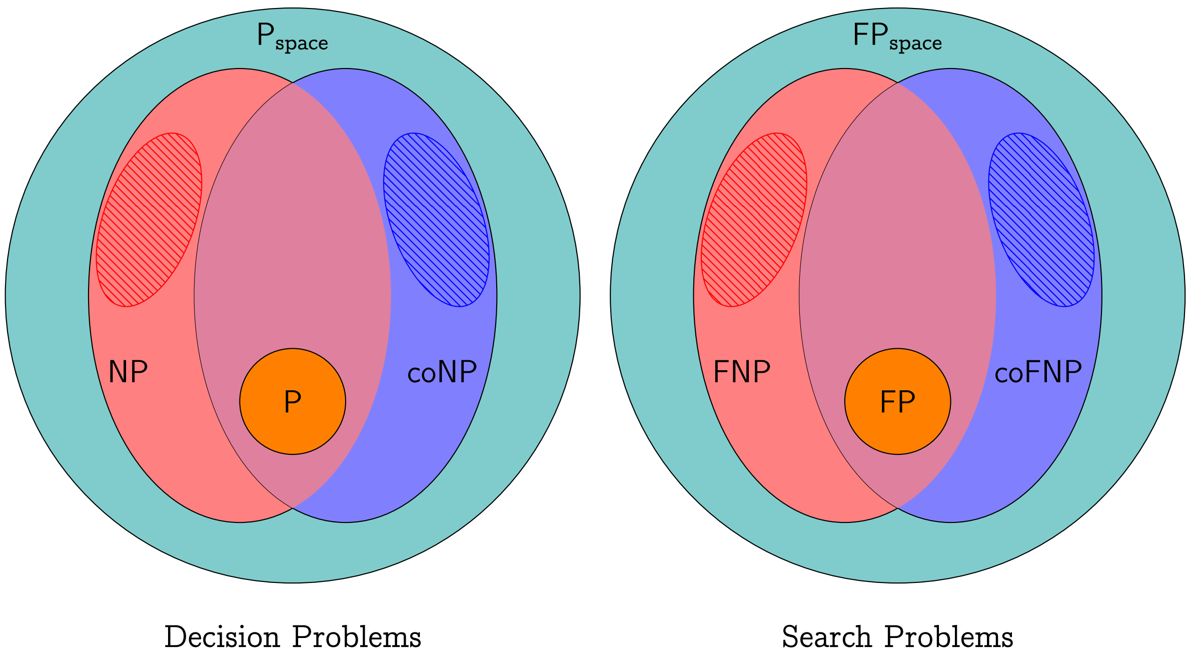


Figure 2.5: Decision problems versus search problems

into a problem in F(NP ∩ coNP) be declaring all instances as "yes"-instances, i.e., by setting $R_{yes} = R$, and $R_{no} = \emptyset$. □

Using this alternative representation of TFNP, it is not hard to see, that TFNP does not contain FNP-complete problems unless P = coNP.

Theorem 2.27 (Johnson et al. (1988); Megiddo and Papadimitriou (1991)). There is an FNP-complete problem in TFNP if and only if NP = coNP.

Proof.
"$\Leftarrow$": if NP = coNP, then, by Theorem 2.26, TFNP = FNP.
"$\Rightarrow$": Let $\Pi_R \in$ TFNP be a FNP-complete problem. In particular, FSAT reduces to Π_R via f and g, i.e., $\text{FSAT}(I, g(x)) \Leftrightarrow R(f(I), x)$. Thus, x is a short "no"-certificate for any unsatisfiable formula I. □

As a consequence, we cannot hope to prove that FNP-completeness of NASH (without proving at the same time the equivalence P = coNP, which is generally considered to be unlikely). In the following, we will identify a subclass of TFNP that is based on a particular proof style that establishes the completeness of the realtion R underlying all its problems Π_R.

2.5.4 The Class PPAD

For (non-degenerate) bimatrix games, we have seen three different proofs for the existence of mixed Nash equilibria, i.e.,

- Nash's first proof based on Kakutani's fixed point theorem (we applied to to a more general class of games in Theorem 1.17),
- Nash's second proof based on Brouwer's fixed point theorem (Theorem 1.14),
- the fact that the Lemke-Howson-Algorithm terminates and computes a Nash equilibrium (Theorem 2.11).

Let us first consider the third proof via the Lemke-Howson-Algorithm. The fact that the algorithm terminates was established via the following simple graph theoretic statement.

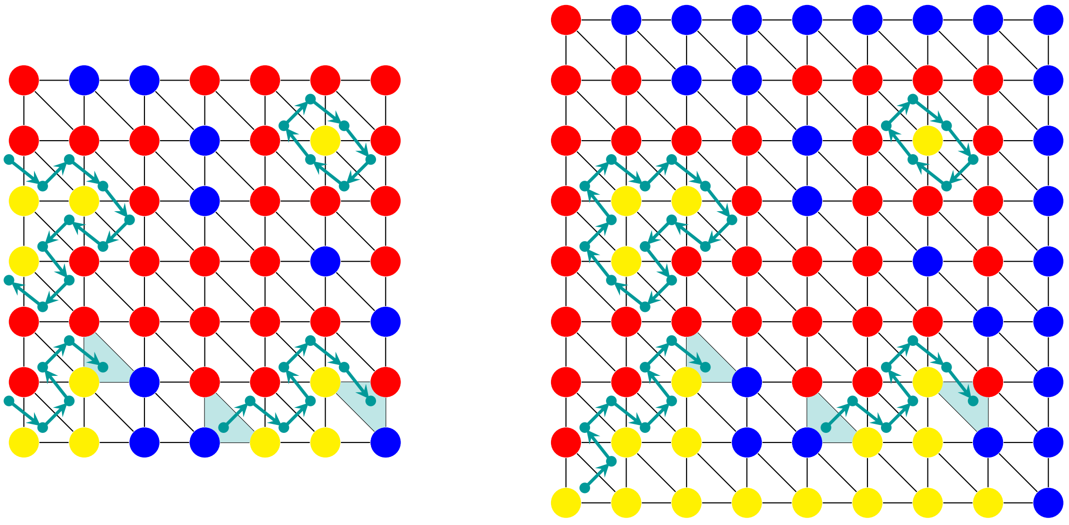
> Every directed graph with in- and outdegree at most one has an equal number of sources and sinks.

The existence of a Nash equilibrium was then established by the mere existence of the artificial node 0, which is a starting node of a path. Since all endpoints of a path (distinct from 0) are a Nash equilibrium, and since there is at least one such endpoint, the existence of a Nash equilibrium follows.
Let us denote the subclass of problems in TFNP whose solutions appear as the (non-trivial) end-points of an implicitly given graph by PPAD. The acronym PPAD stands for polynomial parity argument, directed case.

Definition 2.28 (PPAD). A problem $\Pi_R \in$ PPAD, if, for all instances I, there are polynomial algorithms $B_I, F_I : S_I \to S_I \cup \emptyset$, $S_I = \Sigma^{p(|I|)}$ such that
- $x = B_I(y) \Leftrightarrow y = F_I(x)$
- $B_I(0) = \emptyset$
- $(I, x) \in R \Leftrightarrow |B_I(x) + F_I(x) = 1|$.

The following problem is just a slightly more handy variant of the definition of PPAD. In fact, an alternative definition of PPAD would simply say that PPAD is the set of problems that is reducible to End-of-the-Line.
End-of-the-Line
Input: Boolean circuits encoding functions $B, F : \{0,1\}^n \to \{0,1\}^n$ with $B(0) = 0$.
Output: Either $x \in \{0,1\}^n$ with $B(F(x)) \neq x$, or $x \in \{0,1\}^n$ with $F(x) = x$.

The class PPAD also contains less artificial problems, e.g., the problem to find a panchromatic simplex for a legal coloring of a simplexification of some Eucidean space of fixed dimension. A simplexification of the space is a complete packing with simplices. For the plana, e.g., a simplexification is given in Figure 2.6a. In the plane, a colouring is legal if the colours are red, blue and yellow, all nodes in the leftmost column are not blue, all nodes in the rightmost column are not yellow, all nodes in the uppermost row are not yellow, and all nodes in the lowermost row are not red.
Sperner
Input: Boolean circuit C coloring of $\{0, n\}^d$.



(a) Original triangulation (b) Augmented triangulation

Figure 2.6: Every legal coloring of a triangulation of a the unit square has an odd number of trichromatic triangles. **(a)** shows the original construction to show this result. In **(b)** a border of red nodes is added to the left, a border of yellow nodes to the bottom, and a border of blue nodes to the right and top. As a consequence, one can follow the yellow-red entrance on the lower left, and the problem if computing a tri-chromatic triangle is in PPAD.

OUTPUT: Indices of corners of a panchromatic simplex, or indices of a violation of the legal coloring.

The following theorem states that SPERNER is PPAD-complete, i.e., it is as hard as any problem in PPAD.

Theorem 2.29 (Papadimitriou, 1994; Chen and Deng, 2006). SPERNER is PPAD-complete in dimension $d \geq 2$.

Proof. We here only sketch the membership in PPAD. Consider the instance given in Figure 2.6a. As the coloring is legal the, leftmost column only contains red and yellow nodes. Moreover, since the lowermost row contains no red node and the uppermost row contains no yellow, we known that the node in the lower left is yellow and the node in the upper left is red. Thus, there is an odd number of edges in the leftmost column that connects a red node with a yellow node. Consider an edge that has a red node on top and a yellow node on bottom. Entering this edge, we arrive at a triangle which has at least the colors red and yellow. If the third node is blue, there is nothing left to show. Otherwise, the triangle still has en edge with a red node on the left and a yellow node on the right and we may proceed. Following this procedure, we eventually either find a panchromatic tringle or leave the whole triangulation. Because the coloring is legal, the triangluation may only be left on the leftmost column in which case such a path removes two of the changes from red to yellow. Because the total number of changes is odd, there is still an edge to proceed.

Note that the above argumentation proves the existence of a panchromatic triangle, but not yet the membership in PPAD. For the membership in PPAD it is necessary to have a local algorithm that decides how to proceed on a path. This can be done by the above procedure for trinagles in the inner of the triangulation, but is non-trivial for triangles on the border of the triangulation, where a path leaves the triangulation. We can, however, add a border of red, yellow and blue nodes as in Figure 2.6b to prevents that such a situation occurs. □

Using that Sperner is in PPAD, one can also show that the problem to compute an approximate fixed point (whose existence is guaranteed by Brouwers fixed point theorem) is in PPAD.

Brouwer

Input: polynomial algorithm B evaluating a continuous function $f : [0,1]^n \to [0,1]^n$, approximation requirement ϵ, Lipschitz constant L.

Output: vector $x \in [0,1]^n$ such that $|f(x)-x| \leq \epsilon$, or a pair x, y of points violating Lipschitz constant, i.e., $|f(x)-f(y)| > L|x-y|$, or a point mapped outside of $[0,1]^n$.

Theorem 2.30 (Papadimitriou, 1994; Chen and Deng, 2006). Brouwer is PPAD-complete in dimension $d \geq 2$.

Proof. Again, we here only sketch the membership in PPAD for dimension 2. First, let $\epsilon > 0$ be given. Choose the triangulation of the unit square such that the maximum distance δ between two points of the same triangle is such that $L \cdot \delta + \delta \leq \epsilon$. The main idea of the proof is to color the vertices according to the direction of $f(x) - x$ with the color scheme in Figure 2.7. It is easy to see that this gives a valid coloring of the triangulation of the square provided that the image of f is $[0,1]^2$.

We claim that the yellow corner of a trichromatic triangle is approximate fixed point. To see this, let $x^y = (x_1^y, x_2^y)^\top$, $x^r = (x_1^b, x_2^b)^\top$, $x^b = (x_1^r, x_2^r)^\top$ be the yellow, blue and red corner of the trichromatic triangle, respectively. Since all norms are equivalent in finite dimension, it suffices to show the result for the maximum norm.

Since $\big(f(x^y) - (x^y)\big)^\top e_1$ and $\big(f(x^b) - (x^b)\big)^\top e_1$ have different signs, we obtain

$$\begin{aligned}\Big|\big(f(x^y) - x^y\big)^\top e_1\Big| &\leq \Big|\big(f(x^y) - x^y\big)^\top e_1 - \big(f(x^b) - x^b\big)^\top e_1\Big| \\ &\leq |\big(f(x^y) - f(x^b)\big)^\top e_1| + |\big(x^b - x^y\big)^\top e_1| \\ &\leq L \cdot \delta + \delta.\end{aligned}$$

Similarly, since $\left(f(x^y) - x^y\right)^\top e_2$ and $\left(f(x^r) - x^r\right)^\top e_2$ have different signs,

$$\begin{aligned}\left|\left(f(x^y) - x^y\right)\right)^\top e_2\right| &\leq \left|\left(f(x^y) - x^y\right)^\top e_2 - \left(f(x^r) - x^r\right)^\top e_2\right| \\ &\leq L \cdot \delta + \delta,\end{aligned}$$

which establishes the result. $\square$

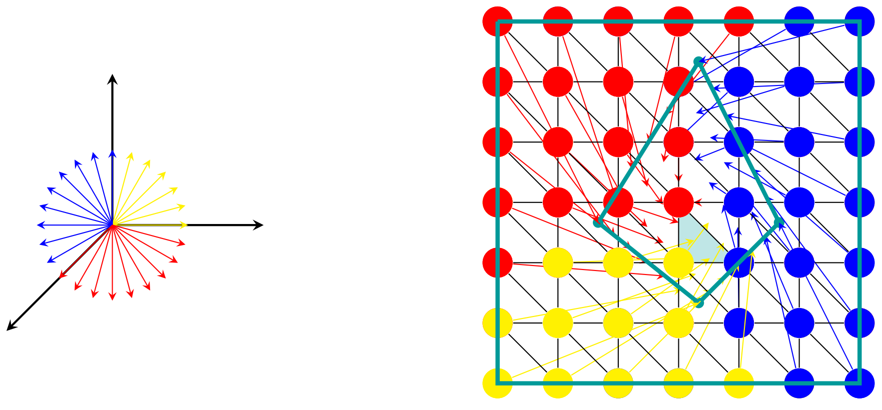
Using Brouwer it is straightforward to show that the following problem to compute an approximate Nash equilibrium is in PPAD.

APPROXIMATE-NASH

INPUT: n number of players, strategy sets $S_1, \ldots, S_n$, utility functions $u_i : S \to \mathbb{N}$, approximation requirement ϵ.

OUTPUT: ϵ-approximate Nash equilibrium.

Theorem 2.31 (Chen et al. (2009)). NASH is PPAD-complete; APPROXIMATE-NASH is PPAD-complete, even for two players.



(a) **(b)**

Figure 2.7: Visualization of reduction of BROUWER to SPERNER. **(a)** General coloring scheme. **(b)** Application of the coloring scheme to a function that maps the outer square continuously to the inner rectangle. As shown in Theorem 2.30, the yellow point of the panchromatic triangle is an approximate fixed point.

2.6 Bibliographic Notes

The Lemke-Howson-Algorithm is due to Lemke and Howson (1964). The presentation in this section and in particular Example 2.2 are based on von Stengel (2007). Savani and von Stengel (2006) showed that the Lemke-Howson-Algorithm

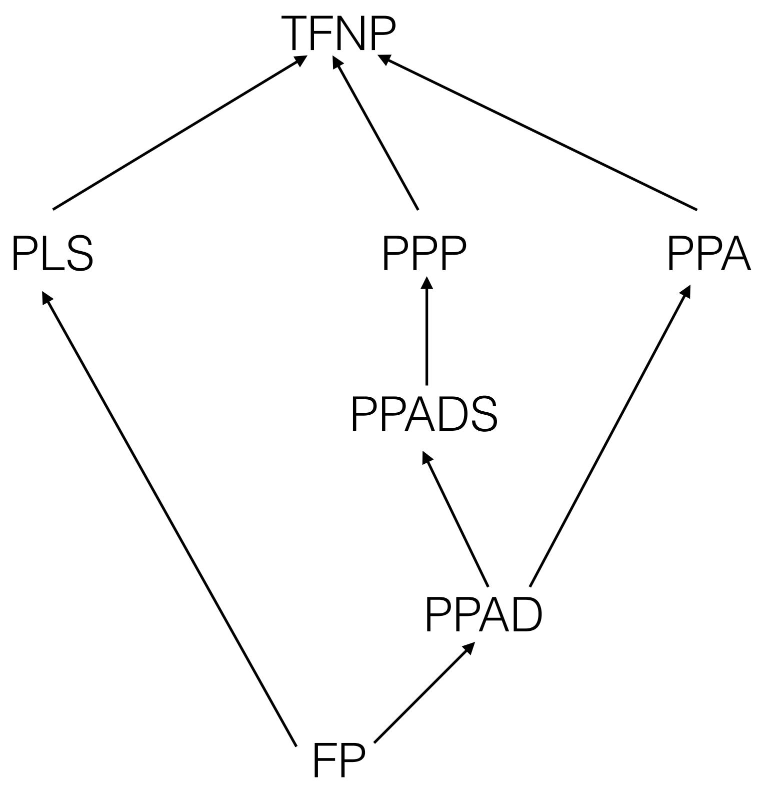


Figure 2.8: Relationship between Subclasses of TFNP.

may take exponential time (Theorem 2.13). Goldberg et al. (2013) strengthened this result as they showed that it P_{space}-complete to compute any of the equilibria that are computed by the Lemke-Howson-Algorithm.

The complexity class PPAD was introduced by Papadimitriou (1994). He showed that Sperner, Brouwer and Kakutani, a natural search problem related to Kakutani's fixed point theorem, are contained in PPAD. He also introduced several variants of the parity argument. If the in- and outdegrees of the nodes are arbitrary, a node with more outgoing than ingoing edges is given, and another unbalanced node is sought, we still obtain PPAD. For other slight variations of the problem, it is not known whether they can be reduced to PPAD. E.g., in the definition of PPAD both sinks and (non-standard) sources are solutions. Requiring that a sink is returned, we obtain the complexity class PPADS. There are some artificial PPADS-problems (Friedl et al., 2009), but no natural problem is known to be contained in PPADS, but not in its subclass PPAD. When we do not know the orientation of the graph, we obtain PPA. More formally, there is an algorithm N_I with $N_I \in \{1, 2\}$, $|N_I(0)| = 1$ and $x \in N_I(y) \Leftrightarrow y \in N_I(x)$ and we seek another solution x with $N_I(x) = 1$. Finding a second Hamiltonian cycle in a cubic graph is contained in PPA. It is open whether this problem is PPA-complete. As for PPAD, there are artificial problems that are complete for PPA (Friedl et al., 2009). A further generalization of PPA is the complexity class PPP which stands for polynomial pidgeonhole principle. For this class, there is a single algorithm N_I and a solution is a vector x with $N_I(x) = 0$ or two vectors x, y with $N_I(x) = N_I(y)$. The natural PPP-complete problem Pigeonhole Circuit; no other problem is known to be PPP-complete. Another problem in PPP is EqualSums. Here, we are given n natural numbers and and we seek two subsets that have the same sum modulo 2^n or a subset whose sum modulo 2^n is 0. It is open whether EqualSums is PPP-complete.

If we change the definition of PPAD so to we require not any sink but the sink reached from the standard source $\mathbf{0}$, we obtain $\mathsf{FP}_{\mathsf{space}}$, the function equivalent of $\mathsf{P}_{\mathsf{space}}$. To the proof of this result is due to Papadimitriou (1994) and uses that any problem in $\mathsf{FP}_{\mathsf{space}}$ can be solved by a polynomial-space-bounded and reversible Turing machine (Bennett, 1989). In fact, also more specialized problems are already $\mathsf{P}_{\mathsf{space}}$-complete, e.g., computing the Nash equilibria found by the algorithm of Lemke and Howson (Goldberg et al., 2013), or computing the solution found by simplex under Danzig's pivot (Fearnley and Savani, 2015).