Complexity of the Gale String Problem for Equilibrium Computation in Games

Marta Maria Casetti

Thesis submitted to the Department of Mathematics
London School of Economics and Political Science
for the degree of Master of Philosophy

London, May 2015

Declaration

I certify that this thesis I have presented for examination for the MPhil degree of the London School of Economics and Political Science is based on joint work with Julian Merschen and Bernhard von Stengel, published in [3].

Abstract

This thesis presents a report on original research, published as joint work with Merschen and von Stengel in *Electronic Notes in Discrete Mathematics* [3]. Our result shows a polynomial time algorithm to solve two problems related to labeled Gale strings, a combinatorial structure introduced by Gale in [9] that can be used in the representation of a particular class of games.

These games were used by Savani and von Stengel [22] as an example of exponential running time for the classical Lemke-Howson algorithm to find a Nash equilibrium of a bimatrix game [13]. It was therefore conjectured that solving these games was a problem complete in the class **PPAD** (Proof by Parity Argument, Directed version); a major motivation for the definition of this class by Papadimitriou [21] was, in turn, to capture the pivoting technique of many results related to the Nash equilibrium, including the Lemke-Howson algorithm.

Our result, on the contrary, sets apart this class of games as a case for which there a Nash equilibrium can be found in polynomial time. Since Daskalakis, Goldberg and Papaditrimiou [5] and Chen and Deng [4] proved the **PPAD**-completeness of finding a Nash equilibrium in general normal-form games, we have a special class of games, unless **PPAD** = **P**.

Our proof exploits two results: first of all, the representation of the Nash equilibria of these games as Gale strings, as seen in Savani and von Stengel [22] [23]; then, we use the polynomial-time solvability of the problem of finding a perfect matching in a graph, proven by Edmonds [6].

Merschen [16] and Végh and von Stengel [25] expanded our ideas to prove further results concerning the index of a Nash equilibrium, as defined in Shapley [24], in the framework of the oiks by Edmonds [7] and Edmonds and Sanità [8].

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis centers on a problem in the field of *algorithmic game theory*, which concerns the study of strategic interactions from the point of view of computer science. Our result consists in an algorithm to find a Nash equilibrium in polynomial time for games in a special class, called Gale games.

Gale games can be represented through a combinatorial structure called Gale strings (Gale [9]), using a construction (Savani and von Stengel [22]) on labeled polytopes derived by the game itself (Lemke and Howson [13], Shapley [24]). This connection was used to build games for which the classic Lemke-Howson Algorithm (Lemke and Howson [13]) takes exponential running time to find a Nash equilibrium (Savani and von Stengel [22]). The Lemke-Howson Algorithm gives a quite straightforward proof of how finding a Nash equilibrium of any two-player game is a problem in the class **PPAD** (Proof By Parity Argument, Directed version; Papadimitriou [21]); its **PPAD**-completeness has been proven (Daskalakis, Goldberg and Papadimitriou [5], Chen and Deng [4]), but the result required the use of approximated ϵ-Nash equilibria. We conjectured that the combinatorial representation of Gale games could be exploited to give an alternative and purely discrete proof of **PPAD**-completeness. Eventually, our result turned out to point in the opposite direction: unless $\mathbf{PPAD} = \mathbf{P}$, Gale games are indeed a case apart, but because of their tractability, not of

their hardness.

The remainder of this chapter will cover some basic definitions and notation that will be used throughout the thesis, concerning polytopes (see Ziegler [29] for details), basic game theory (see Osborne and Rubinstein [19]) and computational complexity (see Papadimitriou [20]). Chapter 2 will deal with the geometric and combinatorial constructions leading to the definition of Gale games: in the first section we will see Nash equilibria of 2-player games as facets or vertices of polytopes built from the game itself; in the second section we will define Gale games, for which these polytopes belong to a special class, and we will translate the framework of the previous section to the point of view of Gale strings. In Chapter 3 we will tackle the computational complexity of the issues arising from the previous chapter: after an introduction to the classes related to proofs by parity argument, we will present different versions of the Lemke-Howson algorithm; finally, in the last section, we will present our original result, solving Gale games in polynomial time. A last chapter will relate further results in the field and open problems.

1.1 Vectors and Polytopes

We will often need to refer to sets of natural numbers; we follow the notation $[n] = \{i \in \mathbb{N} \mid 1 \leq i \leq n\}$.

We denote the transpose of a matrix A as $A^\top$. Vectors will be considered as column vectors, so $u^\top v$ is the scalar product of $u, v \in \mathbb{R}^d$. A vector for which all components are 0's will be denoted as $\mathbf{0}$; a vectors for which all components are 1's will be denoted as $\mathbf{1}$; the i-th component of the *i-th unit vector* e_i is equal to 1, whereas all its other components are 0. An inequality of the form $u \geq v$, is intended to hold for every component.

An *affine combination* of points in an Euclidean space $z_i \in \mathbb{R}^d$, where $i \in [m]$ is $\sum_{i\in[m]} \lambda_i z_i$ where $\lambda_i \in \mathbb{R}$ such that $\sum_{i\in[m]} \lambda_i = 1$; the points z_i are *affinely independent* if none of them is an affine combination of the others.

The *convex hull* of the points z_i is an affine combination with $\lambda_i \geq 0$ for all $i \in [m]$; we denote it as

$$\text{conv}\{z_i \mid i \in [m]\} = \{\sum_i \lambda_i z_i \mid \lambda_i \geq 0, \sum_i \lambda_i = 1 \text{ for } i \in [m]\}.$$

A set of points Z is *convex* if $Z = \text{conv}(Z)$; it has *dimension* d if it has exactly $d+1$ affinely independent points. A convex set of $d+1$ points of dimension d is called a *d-simplex*. The *standard d-simplex* is $\Delta_d = \text{conv}\{e_i \mid i \in [d+1]\}$.

A *polytope* is the convex hull of a finite set of points, not necessarily affinely independent; its *dimension* is its dimension as convex hull. A *polyhedron* is the intersection of finitely many closed halfspaces $\{x \in \mathbb{R}^d \mid a^T x \leq a_0\}$; notice that a bounded polyhedron is a polytope. A *vertex* of a d-dimensional polytope $P = \text{conv}(Z)$ is a point $z \in Z$ such that $\text{conv}(Z \setminus \{z\}) \neq P$; an *edge* of P is a 1-dimensional line segment that has two vertices as endpoints. A *facet* of P is the convex hull of a set of d vertices that lie on a hyperplane of the form $\{x \in \mathbb{R}^d \mid a^T x = a_0\}$ so that $a^T u < a_0$ for all other vertices u of P. A d-dimensional polytope P is *simplicial* if it is the convex hull of a set of at least $d+1$ points $v \in \mathbb{R}^d$ such that no $d+1$ of them are on a common hyperplane; this is equivalent to requiring that every facet of P is a d-simplex. A d-dimensional polytope P is *simple* if every point of P lies on at most than d facets; notice that the points lying on exactly d facets are exactly the vertices. The *polar* of the polytope P is P^Δ, where

$$P = \{x \in \mathbb{R}^d \mid x^\top c_i \leq 1,\ c_i \in \mathbb{R}^d,\ i \in [k]\}$$

$$P^\Delta = \text{conv}\{c_i \mid i \in [k]\}.$$

If P is simplicial or it is simple and it contains the origin $\mathbf{0}$, then $P^{\Delta\Delta} = P$; furthermore, if P is simplicial P^Δ is simple, and vice versa.

1.2 Normal Form Games and Nash Equilibria

The idea of *game* as model of strategic interaction was first introduced by von Neumann [26]. A *finite normal form game* is $\Gamma = (P, S, u)$ where both P and S are finite. The former is the set of *players*; $S = \times_{p \in P}$ is the set of *pure strategy profiles*, where S_p is the set of *pure strategies* of player p. The purpose of each player $p \in P$ is to maximize their *payoff function* $u^p : S \to \mathbb{R}$ and $u = \times_{p \in P} u^p$. By "game" we will always mean "finite normal form game." A *mixed strategy* of player p is a probability distribution on S_p; it can be described as a point on the $(|S_p| - 1)$-dimensional *mixed strategy simplex*

$$\Delta_p = \{x \in \mathbb{R}^{|S_p|} \mid x \geq \mathbf{0},\ \mathbf{1}^\top x = 1\}.$$

The set of *mixed strategy profiles* is the simplicial polytope $\Delta = \times_{p \in P} \Delta_p$; we extend the payoff functions to $u^p : \Delta \to \mathbb{R}$ linearly. A *Nash equilibrium* of a game is a strategy profile in which each player cannot improve their expected payoff by unilaterally changing their strategy; such a strategy is called a *best response*. Note that applying an affine transformation to all the payoffs does not change the Nash equilibria of the game.

Proposition 1.1. *A mixed strategy $x \in \Delta_p$ is a best response against some mixed strategy profile $y \in \times_{q \neq p} \Delta_q$ of the other players if and only if every pure strategy $s_i \in S_p$ chosen with positive probability in x is a best response to y.*

The existence of a Nash equilibrium is guaranteed by the fundamental theorem by Nash ([18]). Notice that there can be more than one equilibrium.

Theorem 1. (Nash [18]) *Every finite game in normal form has at least one Nash equilibrium.*

Bimatrix games are games with only two players; they can be characterized by the $m \times n$ payoff matrices A and B, where a_{ij} and b_{ij} are the payoffs of respectively player 1 and of player 2 when the former plays her ith pure

strategy and the latter plays his jth pure strategy. A bimatrix game is *zero-sum* if $B = -A$, and *symmetric* if $B = A^\top$. We give two classic examples of bimatrix games: the prisoners' dilemma and the coordination game.

Example 1.1. In the symmetric non zero-sum *prisoners' dilemma* of Table 1.1, each player must decide whether to "help" the other one or to "betray" them. If both players help each other, they will get a small reward; if both betray, they will pay a small penalty; if one betrays and the other cooperate the former will get a large reward and the latter will pay a large penalty. The only equilibrium is the profile in which both players betray. If player 2 betrays, the best response of player 1 is to betray, since it gives her payoff 1 instead of 0; if player 2 helps, her payoff for betraying is 3 and her payoff for helping is 2, so betraying is again the best response. The same holds for player 2, so at the equilibrium both players will betray.

1 \ 2	betray	help
betray	1 1	0 3
help	3 0	2 2

Table 1.1 The prisoners' dilemma.

Table 1.2 shows a *coordination* game. Both players drive on a mountain road; they lose if drive on the same side of the road and win if they avoid each other, regardless of which side they take. The pure strategy Nash equilibria are (mountain,valley) and (valley,mountain); there is also a symmetric equilibrium in mixed strategies at $((1/2, 1/2), (1/2, 1/2))$.

1 \ 2	mountain	valley
mountain	0 0	1 1
valley	1 1	0 0

Table 1.2 A coordination game.

1.3 Problems and Complexity

A *deterministic Turing machine* $\mathcal{M}$ (we will imply the "deterministic" from now on) is a representation of an *algorithm* that takes an *input*, runs a *program* manipulating the input, and either does not come to a *halting state* h or it returns an *output*; the latter can be the *accepting state* YES (the Turing machine *accepts* the input), the *rejecting state* NO (the Turing machine *rejects* the input), or a string $\mathcal{M}(x)$. Formally, $\mathcal{M} = (K, \Sigma, \delta, s)$, where the finite set K is the set of *states*; Σ is a finite set of *symbols* (the *alphabet* of $\mathcal{M}$) such that $\Sigma \cap K = \varnothing$ and Σ always contains the symbols $\sqcup$ (*blank*) and $\triangleright$ (*first symbol*). The *transition function* δ is

$$\delta : K \times \Sigma \longrightarrow (K \cup \{h,\ \textsc{Yes},\ \textsc{No}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}.$$

where $\{\leftarrow, \rightarrow, -\} \nsubseteq (K \cup \Sigma)$ correspond to the *cursor directions* "left," "right" and "stay."

A *language* is a set of *strings* of symbols $L \subseteq (\Sigma \setminus \{\sqcup\})^*$; a *class* is a set of languages. A Turing machine $\mathcal{M}$ *decides* L if for every $x \in (\Sigma \setminus \{\sqcup\})^*$ either $\mathcal{M}(x) = \textsc{Yes}$ if $x \in L$ or $\mathcal{M}(x) = \textsc{No}$ if $x \notin L$. If for every $x \in (\Sigma \setminus \{\sqcup\})^*$ either $\mathcal{M}(x) = \textsc{Yes}$, if and only if $x \in L$, or $\mathcal{M}(x)$ does not halt, if and only if $x \notin L$, the Turing machine $\mathcal{M}$ *accepts* L. Finally, $\mathcal{M}$ *computes* a function $f : (\Sigma \setminus \{\sqcup\})^* \to \Sigma^*$ if $\mathcal{M}(x) = f(x)$ for every $x \in (\Sigma \setminus \{\sqcup\})^*$.

An *instance* is the specific input of a problem. A problem P that outputs either YES or NO is a *decision problem*; its *complement* is the problem $\overline{P}$ that

outputs "NO" for each instance of P that outputs "YES", and vice versa; a *function problem* outputs a string more generic than YES or NO An instance x of a *search problem* either returns a string y that satisfies a given relation $R(x, y)$ or it rejects the input if it's not possible to find any such string; if y is guaranteed to exist, the problem is a *total function problem.* By Theorem 1, the problem n-Nash of Table 1.3 is a total function problem.

n-NASH
input : A n-player game.
output: A Nash equilibrium of the game.

Table 1.3 The problem n-NASH.

Let P_1 be a problem and let x be an instance of P_1 that is encoded in $|x|$ bits. P_1 *reduces to the problem* P_2 *in polynomial time* if there exists a function $f : \{0,1\}^* \to \{0,1\}^*$ and a Turing machine $\mathcal{M}$ such that for all $x \in \{0,1\}^*$

1. $x \in P_1 \iff f(x) \in P_2$;
2. $\mathcal{M}$ computes $f(x)$;
3. $\mathcal{M}$ stops after $p(|x|)$ steps, where p is a polynomial.

For any class C of decision problems, the class of all complements of the problems in C is the *complement class* co $-$ C. A problem P is *hard* for a class C if every problem in C is polynomial-time reducible to P; that is, if P is hard to solve at least as every problem in C. A *complete* for the class C is a C$-$*hard* problem that is also in C. Intuitively, if P_1 is polynomial-time reducible to P_2, it takes polynomial time to "translate" P_1 to P_2, and then to "translate back" a solution of P_2 as a solution of P_1. This is particularly useful if P_2 is hard; then the problem P_1 is at least as "difficult"; if P_2 is also complete, it can be used as a test of belonging to its class.

The complexity class **P** contains all the *polynomially decidable problems*; that is, all problems P such that there exists a Turing machine $\mathcal{M}$ that outputs

either YES or NO for all inputs $x \in \{0,1\}^*$ of P after $p(|x|)$ steps, where p is a polynomial. A problem P belongs to the class **NP**, *non-deterministic polynomial-time problems*, if there exists a Turing machine $\mathcal{M}$ and polynomials p_1, p_2 such that

1. for all $x \in P$ there exists a *certificate* $y \in \{0,1\}^*$ such that $|y| \leq p_1(|x|)$;
2. $\mathcal{M}$ accepts the combined input xy, stopping after at most $p_2(|x| + |y|)$ steps;
3. for all $x \notin P$ there does not exist $y \in \{0,1\}^*$ such that $\mathcal{M}$ accepts the combined input xy.

Informally, a decision problem is in **P** if the answer to its question can be found in a number of steps that is polynomial in the input of the problem, and a decision problem is in **NP** if it takes polynomial time to verify whether the "certificate solution" y is, indeed, a correct answer to the question posed by the problem. The class #**P** is the class of all problems that output the number of possible certificates for a problem in **NP**. Finally, the class **PSPACE** is the set of decision problem that are solved by a Turing machine requiring a polynomial amount of space of the input size n. The relation $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$ holds, with strict inclusions being an open problem; it is also still unknown if $\mathbf{NP} = \mathbf{co\text{-}NP}$.

Chapter 2

Labels, Polytopes and Gale Strings

In the remainder of this thesis we will focus bimatrix games. We will identify the game with its payoff matrices (A, B), and we will assume that both A and B are non-negative, and that neither A nor $B^\top$ has a zero column; this can be done without loss of generality, applying an affine transformation to A and B if necessary.

We will start by showing a construction to represent a bimatrix game to two simplices subdivided in labeled regions such that the Nash equilibria of the game correspond to "completely labeled" points of the simplices. This idea is due to Lemke and Howson [13]; we will follow the very clear approach given by Shapley [24]. From labeled regions we will then move on to an equivalent formulation in terms of polytopes; a result by McLennan and Tourky [14] on a special case of games, called imitation games, and its extension to the more general unit vector games, due to Balthasar [1] and Savani and von Stengel [23], will allow us to simplify this construction. The problem 2-NASH will then be reduced to a problem on the facets or vertices of a labeled polytope, under some simple conditions.

In the second section of the chapter we will restrict our scope to Gale games,

a case of unit vector games for which the polytopes in the previous section can be represented through a combinatorial structures called Gale strings. We will first show the correspondence between polytopes and Gale strings, then we will define a labeling for the latter that will lead to a reduction from 2-NASH to a problem on Gale strings.

2.1 Bimatrix Games and Labels

Let $n, m \in \mathbb{N}$ with $m \leq n$. A *labeling* of a set X is a function $l : X \to [m]$; an n-uple $x = (x_1, \ldots, x_n) \in X^n$ is *completely labeled* if each label $j \in [m]$ appears in $(l(x_1), \ldots, l(x_n))$; formally, if $\{j \in [m] \mid l(x_i) = j \text{ for some } i \in [n]\} = [m]$.

Let (A, B) be bimatrix game, and let X and Y be the mixed strategy simplices of respectively player 1 and 2, that is

$$X = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ \mathbf{1}^\top x = 1\};$$
$$Y = \{y \in \mathbb{R}^n \mid y \geq \mathbf{0},\ \mathbf{1}^\top y = 1\}. \tag{2.1}$$

A *labeling* of the game is then given as follows:

1. the m pure strategies of player 1 are denoted as $i \in [m]$;
2. the n pure strategies of player 2 are denoted as $m + j$ with $j \in [n]$;
3. each mixed strategy $x \in X$ of player 1 has
   - label i for each $i \in [m]$ such that $x_i = 0$,
   - label $m + j$ for each $j \in [n]$ such that the j-th pure strategy of player 2 is a best response to x;
4. each mixed strategy $y \in Y$ of player 2 has
   - label $m + j$ for each $j \in [n]$ such that $y_j = 0$,
   - label i for each $i \in [m]$ such that the i-th pure strategy of player 1 is a best response to y.

This labeling can be used to characterize the Nash equilibria of the game.

Theorem 2. (Shapley [24]) *Let $(x, y) \in X \times Y$; then (x, y) is a Nash equilibrium of the bimatrix game (A, B) if and only if (x, y) is completely labeled.*

Proof. The mixed strategy $x \in X$ has label $m + j$ for some $j \in [n]$ if and only if the j-th pure strategy of player 2 is a best response to x; by Proposition 1.1, this is a necessary and sufficient condition for player 2 to play his j-th strategy at every equilibrium where player 1 chooses x. The analogous holds for the strategies $y \in Y$ and player 1 playing her i-th strategy in response to player 2 choosing y. Therefore, at an equilibrium (x, y) all labels $m + j$, with $j \in [n]$, will appear either as labels of x or of y. □

An useful graphical representation of labels on the simplices X and Y is done by labeling the outside of each simplex according to the player's own pure strategies that are not played, and by subdividing its interior in closed polyhedral sets corresponding to the other player's pure best response strategies, called *best response regions*. We give an example of this construction.

Example 2.1. Consider the 3×3 game (A, B) with

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}. \tag{2.2}$$

Figure 2.1 shows the mixed strategy simplices X and Y: the exterior facets are labeled with the pure strategy that is played at the opposite vertex of the simplex; the interior is covered by the best response regions, labeled by to the other player's pure best response strategies. For example, the best-response region in Y with label 1 is the set of all the $(y_1, y_2, y_3) \in \mathbb{R}^3$ such that $y_1 \geq y_2$ and $y_1 \geq y_3$. There is only one pair (x, y) that is completely labeled, namely $x = (\frac{1}{3}, \frac{2}{3}, 0)$, that has labels $3, 4, 5$, and $y = (\frac{1}{2}, \frac{1}{2}, 0)$, that has labels $1, 2, 6$; this gives the only Nash equilibrium of the game.

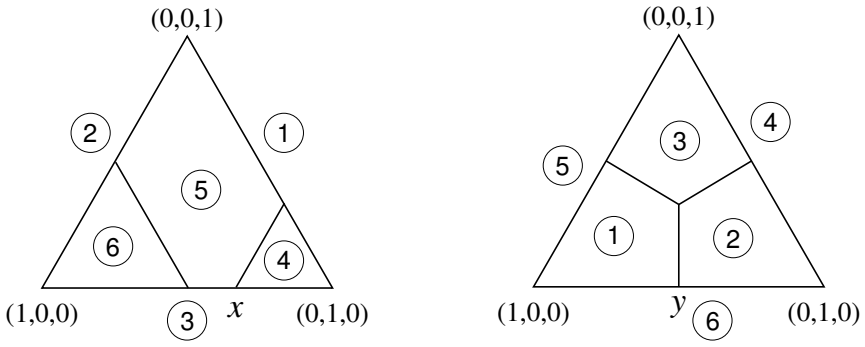


Figure 2.1 The labeled best response regions of the mixed strategy simplices of game (2.2).

The representation of a game and its Nash equilibria in terms of best response regions can be translated to an equivalent construction on polytopes. The first step is to notice that the best-response regions can be obtained as projections on X and Y of the *best-response facets* of the polyhedra

$$\begin{aligned} \overline{P} &= \{(x, v) \in X \times \mathbb{R} \mid B^\top x \leq \mathbf{1}v\}; \\ \overline{Q} &= \{(y, u) \in Y \times \mathbb{R} \mid Ay \leq \mathbf{1}u\}. \end{aligned} \tag{2.3}$$

In $\overline{P}$, these facets are the points $(x, v) \in X \times \mathbb{R}$ such that $(B^\top x)_j = v$, which in turn correspond to the strategies $x \in X$ of player 1 that give exactly payoff v to player 2 when he plays strategy j; the projection of the facet defined by $(B^\top x)_j = v$ to X has then label j. Analogously, the facet of $\overline{Q}$ given by the points $(y, u) \in Y \times \mathbb{R}$ such that $(Ay)_i = u$ projects to the best-response region of Y with label i.

Example 2.2. In Example 2.1, the inequalities $B^\top x \leq \mathbf{1}v$ are

$$\begin{aligned} 3x_2 &\leq v; \\ 2x_1 + 2x_2 + 2x_3 &\leq v \\ 4x_1 &\leq v. \end{aligned}$$

Figure 2.2 shows the best-response facets of $\overline{P}$ and their projection to X by

ignoring the payoff variable v, which gives the subdivision of X into best-response regions of Figure 2.1.

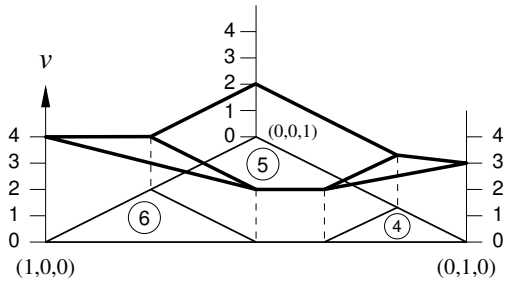


Figure 2.2 The best response polyhedron of game (2.2).

Given the assumptions on non-negativity of A and $B^\top$, we can change coordinates to x_i/v and y_j/u and replace $\overline{P}$ and $\overline{Q}$ with the *best-response polytopes*

$$\begin{aligned} P &= \{x \in \mathbb{R}^m \;\mid\; x \geq \mathbf{0},\; B^\top x \leq \mathbf{1}\}; \\ Q &= \{y \in \mathbb{R}^n \;\mid\; y \geq \mathbf{0},\; Ay \leq \mathbf{1}\}. \end{aligned} \tag{2.4}$$

The polytope P is the intersection the $m+n$ half spaces corresponding to either player 1 avoiding her i-th pure strategy or a best response of player 2 that gives non-zero probability to his j-th strategy, where $i \in [m]$ and $j \in [n]$; the analogous holds for Q. Formally, a point $x \in P$ has label k if and only if either $x_k = 0$ for $k \in [m]$ or $(B^\top x)_k = 0$ for $k \in [n]$, and a point in Q has label k if and only if either $y_k = 0$ for $k \in [n]$ or $(Ay)_k$ for $k \in [m]$. Then a point $(x, y) \in P \times Q$ is completely labeled if and only if it satisfies the *complementarity condition*

$$\begin{aligned} x_i = 0 \quad &\text{or } (Ay)_i = 1 \qquad \text{for all } i \in [m]; \\ y_j = 0 \quad &\text{or } (B^\top x)_j \qquad \text{for all } j \in [n]. \end{aligned} \tag{2.5}$$

Therefore, if $(x, y) \in P \times Q$ is completely labeled either the corresponding point in $\overline{P} \times \overline{Q}$ is a Nash equilibrium or $(x, y) = (\mathbf{0}, \mathbf{0})$; we will refer to the latter case as *artificial equilibrium*.

Example 2.3. Keeping on with Example 2.1, the best response polyhedron $\overline{P}$ of Figure 2.2 becomes the best response polytope of Figure 2.3. Notice that the vertex $(\mathbf{0}, \mathbf{0})$ is completely labeled, since it is labeled by the labels $x_k = 0$ with $k = 1, 2, 3$ in P and $k = 4, 5, 6$ in Q.

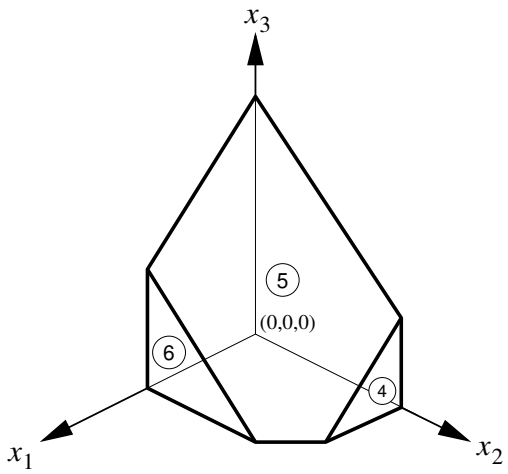


Figure 2.3 The best response polytope P of game (2.2).

We will now see some special cases of games and how they are related to each other in terms of computational complexity. First of all, we note that any bimatrix game can be "symmetrized"; the result is due to Gale, Kuhn and Tucker [10] for zero-sum games, while its extension to non-zero-sum games is a folklore result.

Proposition 2.1. *Let (A, B) be a bimatrix game and let (x, y) be one of its Nash equilibria. Then (z, z), where $z = (x, y)$, is a Nash equilibrium of the symmetric game $(C, C^\top)$, where*

$$C = \begin{pmatrix} 0 & A \\ B^\top & 0 \end{pmatrix}. \tag{2.6}$$

McLennan and Tourky [14] have proven a result in the opposite direction of Proposition 2.1: any symmetric game can be translated into a *imitation game*, where the payoff matrix of player 1 is the identity matrix I. In any Nash

equilibrium of (I, B), the mixed strategy x of player 1 corresponds exactly to the symmetric equilibrium (x, x) in the symmetric game defined by the payoff matrix of player 2. Since it takes polynomial time in the size of a matrix to calculate its transpose, an algorithmm that finds a Nash equilibrium of a bimatrix game can be used to find a symmetric Nash equilibrium of a symmetric game.

Theorem 3. (McLennan and Tourky [14]) *The pair (x, x) is a symmetric Nash equilibrium of the symmetric bimatrix game $(C, C^\top)$ if and only if there is some y such that (x, y) is a Nash equilibrium of the imitation game (I, B) with $B = C^\top$.*

Notice that Theorem 3 applies to the symmetric equilibria of the symmetric game but not to all its Nash equilibria; there could be non-symmetric equilibria of $(C, C^\top)$ that are not found through the imitation game, as shown in the following example.

Example 2.4. As an example, consider the symmetric game $(C, C^\top)$ with

$$C = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 2 & 2 \\ 4 & 0 & 0 \end{pmatrix}, \qquad C^\top = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}; \tag{2.7}$$

the corresponding imitation game is $(I, C^\top) = (A, B)$, seen in Example 2.1. Figure 2.4 shows the labeled mixed-strategy simplices X and Y for the game (2.7); since the game is symmetric, only the labels are different. In addition to the symmetric equilibrium (x, x) where $x = (\frac{1}{3}, \frac{2}{3}, 0)$, the game has two non-symmetric equilibria in (a, b) and (b, a) with $a = (\frac{1}{2}, \frac{1}{2}, 0)$ and $b = (0, \frac{2}{3}, \frac{1}{3})$. The imitation game (A, B), on the other hand, has only one equilibrium (x, y), corresponding to (x, x), with $y = (\frac{1}{2}, \frac{1}{2}, 0)$.

The characterization of Nash equilibria as completely labeled pairs (x, y) holds for arbitrary bimatrix games. From now on, we will impose a further condition: all points in P will have at most m labels, and all points in Q have

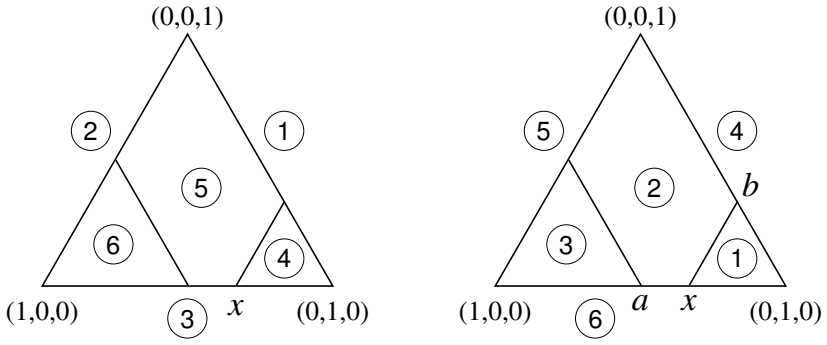


Figure 2.4 The best response regions of the symmetric game (2.7).

at most n labels. These games are called *nondegenerate*; since any game can be made nondegenerate by lexicographic perturbation (see von Stengel [27]), we can impose the nondegeneracy condition without loss of generality. In an equilibrium (x, y) of a nondegenerate game each label appears exactly once; this also means that the number of pure best response strategies against a mixed strategy is never larger than the size of the support of that mixed strategy. Geometrically, this means that no point of the best response polytope P lies on more than m facets and no point of the best response polytope Q lies on more than n facets, so both P and Q are simple. Furthermore, a point of P has exactly m labels if and only if it is a vertex, and a point of Q has exactly n labels if and only if it is a vertex; therefore, all completely labeled points (x, y) are vertices of the best response polytopes, and Nash equilibria are isolated points.

Example 2.5. An example of degenerate game is given by $(C, C^\top)$ with

$$C = \begin{pmatrix} 0 & 4 & 0 \\ 2 & 2 & 2 \\ 4 & 0 & 0 \end{pmatrix}, \qquad C^\top = \begin{pmatrix} 0 & 2 & 4 \\ 4 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}. \tag{2.8}$$

As shown in Figure 2.5, the mixed strategy $x = (\frac{1}{2}, \frac{1}{2}, 0)$, that also defines the unique symmetric equilibrium (x, x) of the game, has three pure best responses; the Nash equilibria (x, y) of the imitation game $(I, C^\top)$ are not unique, since

any convex combination of $(\frac{1}{2}, \frac{1}{2}, 0)$ and $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ can be chosen for y.

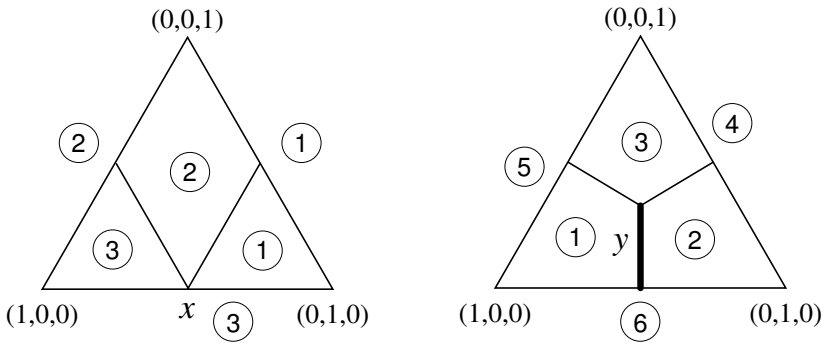


Figure 2.5 Left: The best-response regions of the degenerate symmetric game (2.8). Right: The best-response regions of the imitation game $(I, C^\top)$.

A generalization of imitation games is the class of *unit vector games*, introduced by Balthasar [1]; these are defined as bimatrix games of the form (U, B) where the columns of the matrix U are unit vectors. By the results above, finding a Nash equilibrium of a bimatrix game is at least as hard as finding a Nash equilibrium of a unit vector game. Savani and von Stengel [23] have shown that in unit vector games the problem of finding a completely labeled vertex of the product of the best response polytopes $P \times Q$ can be simplified to the problem of finding a completely labeled vertex of one polytope that encodes all the game.

Theorem 4. (Savani and von Stengel [23]) *Let $l : [n] \to [m]$ be a labeling, and let (U, B) be the unit vector game with $U = (e_{l(1)} \ \cdots \ e_{l(n)})$. Let $N_i = \{j \in [n] \mid l(j) = i\}$ for $i \in [m]$, and consider the polytopes P^l and Q^l*

$$P^l = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}; \tag{2.9}$$
$$Q^l = \{y \in \mathbb{R}^n \mid y \geq \mathbf{0},\ \sum_{\substack{j \in N_i \\ i \in [m]}} y_j \leq 1\}.$$

Let l_f be the labeling of the facets of P^l defined as follows:

$$\begin{aligned} x_i \geq 0 \quad & \text{has label } i \quad \text{for } i \in [m]; \\ (B^\top x)_j \leq 1 \quad & \text{has label } l(j) \quad \text{for } j \in [n]. \end{aligned} \tag{2.10}$$

Then $x \in P^l$ is a completely labeled vertex of $P^l \setminus \{\mathbf{0}\}$ if and only if there is some $y \in Q^l$ such that, after scaling, the pair (x, y) is a Nash equilibrium of (U, B)

Proof. Let P and Q be the best response polytopes of (U, B) as in (2.4), and let $(x, y) \in P \times Q \setminus \{(\mathbf{0}, \mathbf{0})\}$ be a Nash equilibrium of (U, B); then (x, y) is completely labeled in $[m + n]$. If $x_i = 0$, then x has label $i \in m$; if $x_i > 0$, then y has label i, so $(Uy)_i = 1$; therefore for some $j \in [n]$ we have $y_j > 0$ and $U_j = e_i$; that is, we have $y_j > 0$ and $l(j) = i$ for some $j \in [n]$. Since $y_j > 0$, $x \in P$ has label $m + j$; then, $(B^\top x)_j = 1$; therefore $x \in P^l$ has label $l(j) = i$. Hence, x is a completely labeled vertex of P^l.

Conversely, let $x \in P^l \setminus \{\mathbf{0}\}$ be completely labeled. If $x_i > 0$, then there is $j \in [n]$ such that $(B^\top x) = j$ and $l(j) = i$; that is, $j \in N_i$. For all $i \in [m]$ such that $x_i > 0$, define y as follows: $y_j = 1$; $y_h = 0$ for all $h \in N_i \setminus \{j\}$. Then $(x, y) \in P \times Q$ is completely labeled. □

Example 2.6. The game in Example 2.1 is a unit vector game with $l(i) = i$. In the polytope P^l of Figure 2.6 the labels 4, 5 and 6 of the best response polytope P of Figure 2.3 are replaced by 1, 2 and 3, since the corresponding columns of A are the unit vectors e_1, e_2, e_3. The only completely labeled point of P^l are the origin $\mathbf{0}$, corresponding to the "artificial" equilibrium, and x, corresponding to the unique Nash equilibrium of the unit vector game (2.2).

We will now move on the dual version of Theorem 4, as given by Balthasar [1]. We can traslate the polytope P^l of (2.9) to $P = \{x - \mathbf{1} \mid x \in P^l\}$, possibly multiplying all payoffs in B by a constant so that $\mathbf{0}$ is in the interior of P. We

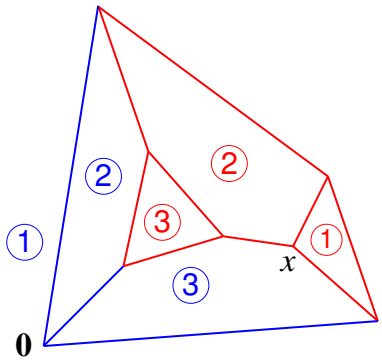


Figure 2.6 The polytope P^l of the unit vector game (2.2).

have that

$$P = \{x + \mathbf{1} \geq \mathbf{0},\ (x+\mathbf{1})^\top B \leq \mathbf{1}\} =$$
$$= \{x \in \mathbb{R}^m \mid -x_i \leq 1 \text{ for } i \in [m],\ x^\top (b_j/(1-\mathbf{1}^\top b_j)) \leq 1 \text{ for } j \in [n]\}.$$

The polar of P is then

$$Q = P^\Delta = \text{conv}(\{e_i \mid i \in [m]\} \ \cup \ \{c_j \mid j \in [n]\}) \tag{2.11}$$

where $c_j = b_j/(1-\mathbf{1}^\top b_j)$. Since P is a simple polytope with $\mathbf{0}$ in its interior, $Q^\Delta = P^{\Delta\Delta} = P$. Furthermore, Q is simplicial and its facets correspond to the vertices of P and vice versa. We label the vertices of Q as the corresponding facets in P^l, so the completely labeled facets of Q correspond to the completely labeled vertices of P^l. In particular, the facet corresponding to $\mathbf{0}$ is

$$\begin{aligned} F_0 &= \{x \in P^\Delta \mid -\mathbf{1}^\top x = 1\} \\ &= \text{conv}\{e_i \mid i \in [m]\}. \end{aligned} \tag{2.12}$$

Theorem 4 then translates into the following.

Theorem 5. (Balthasar [1]) *Let Q be a labeled m-dimensional simplicial polytope with $\mathbf{0}$ in its interior and vertices $e_1, \ldots, e_m, c_1, \ldots, c_n$ such that (2.12) is a facet of Q. Let (U, B) be a unit vector game, with $U = (e_{l(1)} \ \cdots \ e_{l(n)})$ for*

a labeling $l : [n] \to [m]$ *and* $B = (b_1 \ \cdots \ b_n)$*, where* $b_j = c_j/(1 + \mathbf{1}^\top c_j)$ *for* $j \in [n]$*. Let* l_v *be the labeling of the vertices of* Q *given by*

$$\begin{aligned} l_v(-e_i) &= i \qquad \text{for } i \in [m]; \\ l_v(c_j) &= l(j) \qquad \text{for } j \in [n]. \end{aligned} \tag{2.13}$$

Then a facet $F \neq F_0$ *of* Q *with normal vector* v *is completely labeled if and only if* (x, y) *is a Nash equilibrium of* (U, B)*, where* $x = (v + \mathbf{1})/(\mathbf{1}^\top(v + \mathbf{1}))$*, so that* $x_i = 0$ *if and only if* $e_i \in F$ *for* $i \in [m]$ *and the mixed strategy* y *is the uniform distribution on the set of the pure best replies to* x*, which in turn correspond to all* $j \in [n]$ *such that* c_j *is a vertex of* F*.*

Theorem 4 gives a correspondence between completely labeled vertices of P^l and equilibria of the unit vector game (U, B) with the "artificial" equilibrium corresponding to the vertex $\mathbf{0}$; Theorem 5 gives a correspondence between completely labeled facets of Q and equilibria of (U, B) with the "artificial" equilibrium corresponding to the facet F_0 in (2.12).

Given a bimatrix game (A, B), it takes polynomial time to write and solve the linear equations defining its best response polyhedra $\overline{P}, \overline{Q}$ and its best response polytopes P, Q. It also take polynomial time to label $\overline{P}, \overline{Q}$ and P, Q. Analogously, given a unit vector game (U, B), it takes polynomial time to construct and label the polytope P^l and its polar. Therefore, Theorem 4 gives a polynomial time reduction from the problem 2-Nash to the problem Another Completely Labeled Vertex of Table 2.1 and Theorem 5 gives a dual reduction to the problem Another Completely Labeled Facet of Table 2.2.

Another Completely Labeled Vertex	
input :	A m-dimensional simple polytope P with $m+n$ facets; a labeling $l_f : [m+n] \to [n]$; a vertex v_0 of P that is completely labeled by l_f.
output:	A vertex $v \neq v_0$ of P that is completely labeled by l_s.

Table 2.1 The problem Another Completely Labeled Vertex.

Another Completely Labeled Facet	
input :	A simplicial m-dimensional polytope Q with $m+n$ vertices; a labeling $l_v : [m+n] \to [n]$; a facet F_0 of Q that is completely labeled by l_v.
output:	A facet $F \neq F_0$ of Q that is completely labeled by l_v.

Table 2.2 The problem Another Completely Labeled Facet.

Proposition 2.2. 2-Nash *reduces in polynomial time to* Another Completely Labeled Vertex *and* Another Completely Labeled Facet.

2.2 Cyclic Polytopes and Gale Strings

We now apply the results of the previous section to unit vector games for which the best response polytope is the dual of a cyclic polytope. These polytopes are characterized by their representation as a combinatorial structure, called Gale strings. We will first give the definition of cyclic polytope, then of Gale string, then the theorem by Gale [9] that shows the equivalence of the two representations.

The *moment curve* in dimension d is defined as

$$\mu_d : \mathbb{R} \longrightarrow \mathbb{R}^d \qquad \mu_d : t \longmapsto (t, t^2, \ldots, t^d)^\top. \tag{2.14}$$

The *cyclic polytope* in dimension d with n vertices, where $n > d$, is

$$C_d(n) = \text{conv}\{\mu_d(t_i) \mid t_1 < \ldots < t_n \text{ affinely independent }\}. \tag{2.15}$$

Example 2.7. Figure 2.7 shows the cyclic polytope in dimension 3 with 6 facets.

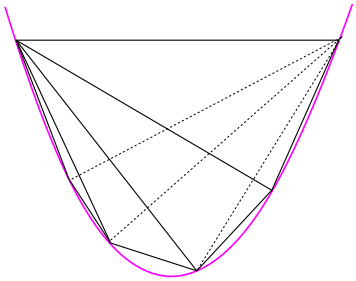


Figure 2.7 The cyclic polytope $C_3(6)$.

Given $k \in \mathbb{N}$ and a set S, we can represent the function $f : [k] \to S$ as the string $s = s(1)s(2)\cdots s(k)$; we have a *bitstring* if $S = \{0, 1\}$. A maximal substring of consecutive 1's in a bitstring is called a *run*; an *interior run* is bounded on both sides by 0's. We will use the notation $\mathbf{1}^k$ for a run of length k and 0^k for a string of 0's of length k. A *Gale string of length n and dimension d*, where $n > d$, is a bitstring $s \in G(d, n)$ satisfying the following conditions:

1. exactly d bits of s are equal to $\mathbf{1}$;
2. (*Gale evenness condition*) $0\mathbf{1}^k0$ is a substring of $s \quad \Rightarrow \quad k$ is even.

In general, the Gale evenness conditions allows for Gale strings that start or end with an odd-length run; if d is even then s can start with an odd run if and only if it ends with an odd run. When d is even, we can therefore see the Gale strings in $G(d, n)$ as "loops" obtained by "glueing together" the endpoints of the strings; then all runs on the loops are even. Formally, we can see the indices of a Gale string $s \in G(d, n)$ with d even as equivalence classes modulo n.

Example 2.8. As an example of even d, we have

$$G(4,6) = \{\mathbf{111}00, \mathbf{111}00\mathbf{1}, \mathbf{11}00\mathbf{11}, \mathbf{1}00\mathbf{111}, 00\mathbf{1111},$$
$$0\mathbf{1111}0, \mathbf{11}0\mathbf{11}0, \mathbf{1}0\mathbf{11}0\mathbf{1}, 0\mathbf{11}0\mathbf{11}\}$$

The strings **111**00, **111**00**1**, **11**00**11**, **1**00**111**, 00**1111** and 0**1111**0 are equivalent under a cyclic shift (if considering the strings as "loops", the **1**'s are all consecutive), as are the strings **11**0**11**0, **1**0**11**0**1** and 0**11**0**11** (two runs of two **1**'s separated by a single 0). As an example of odd d, we have

$$G(3,5) = \{\mathbf{11}100, \mathbf{11}0\mathbf{11}0, \mathbf{1}00\mathbf{11}, \mathbf{11}00\mathbf{1}, 0\mathbf{11}0\mathbf{1}, 00\mathbf{111}\}$$

Notice that 0**1**0**11** is a shift of **1**0**11**0 but it is not a Gale string.

The relation between cyclic polytopes and Gale strings was given by Gale [9].

Theorem 6. (Gale [9]) *For any $d, n \in \mathbb{N}$, where $n > d$,*

$$F \text{ is a } \qquad \text{facet of } C_d(n)$$
$$\iff$$
$$F = \text{conv}\{\mu(t_j) \;\mid\; s(j) = 1 \quad \text{for } s \in G(d,n)\}. \tag{2.16}$$

Proof. By definition,

$$C_d(n) = \text{conv}\{\mu_d(t_j) \;\mid\; t_1 < \ldots < t_n \text{ affinely independent}\}.$$

Let $\overline{t_1} < \ldots < \overline{t_d}$ be a choice of some of the t_j's in the definition of $C_d(n)$. Then the $\mu_d(\overline{t_i})$'s define an hyperplane H that crosses the moment curve $\mu_d(t)$ at all and only the $\overline{t_i}$'s. Given the definition of moment curve, the hyperplane H is never tangent to the moment curve, and every crossing gives a "change of sign"; that is, if $t, t' \notin \{\overline{t_i}\}$ and $t < \overline{t_i} < t'$ for one and only one $\overline{t_i}$ then $\mu_d(t)^\top \mu_d(t') < 0$. A facet F of the cyclic polytope $C_d(n)$ then corresponds to a choice of $\overline{t_i}$'s such that for all the t_k's such that $t_k \notin \{\overline{t_i} \;\mid\; i \in [d]\}$ all

the $\mu_d(t_k)$'s have the same sign. This can happen only if for every couple of t_k's the moment curve has an even number of changes of sign between them; therefore there is an even number of $\overline{t_i}$'s between any two t_k's. Let s be the bitstring in which the $\mathbf{1}$'s correspond to the t_i's and and the 0's correspond to the other t_k's; then the condition for being a facet in (2.16) becomes the Gale evenness condition. □

Notice that the moment curve has exactly d zeroes implies that the each facet of $C_d(n)$ is a d-simplex, so $C_d(n)$ is simplicial and the choice of the t_j's is irrelevant.

Example 2.9. Consider the facet F of the cyclic polytope $C_3(6)$ marked in blue in Figure 2.8. If we label the vertices on the moment curve as t_i, with $i \in [n]$, and we set $s(i) = 1$ if t_i is a vertex of F and $s(i) = 0$ otherwise, we see that the corresponding Gale string $s \in G(3, 6)$ is $s = \mathbf{1}00\mathbf{11}0$. Figure 2.9 shows the intersection of the moment curve and the hyperplane H in $\mathbb{R}^3$ defined by the t_i such that $s(i) = 1$.

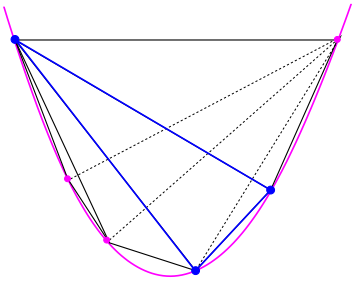


Figure 2.8 The facet of the cyclic polytope $C_3(6)$ corresponding to the Gale string $s = \mathbf{1}00\mathbf{11}0 \in G(3, 6)$.

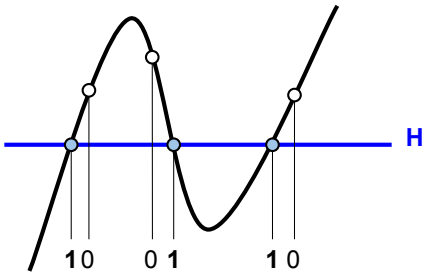


Figure 2.9 The facet of the cyclic polytope $C_3(6)$ given by the intersection of the moment curve and the hyperplane where $\mu_3(t_1) = \mu_3(t_4) = \mu_3(t_5) = 0$ corresponds to the Gale string $s = \mathbf{1}00\mathbf{11}0 \in G(3, 6)$.

Example 2.10. Figure 2.10 shows the cyclic polytope $C_4(6)$, with the exterior facet corresponding to the Gale string $s = \mathbf{1111}00$. Figure 2.11 shows the string $s = \mathbf{1111}00$ as intersection of the moment curve and the hyperplane H.

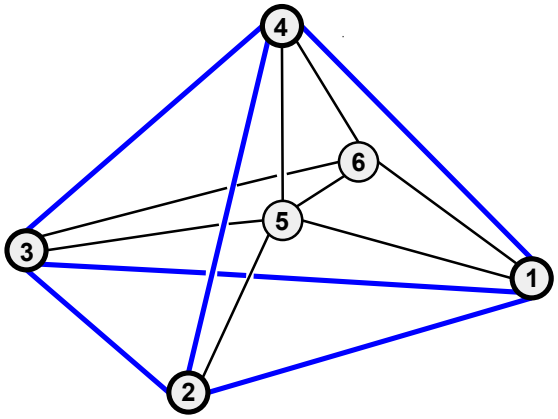


Figure 2.10 The cyclic polytope $C_4(6)$. The thin lines represent the edges inside the exterior facet, in bold lines; the latter corresponds to the Gale string $s = \mathbf{1111}0 \in G(4,6)$.

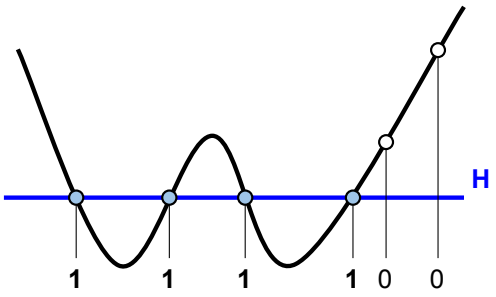


Figure 2.11 The facet of the cyclic polytope $C_4(6)$ given by the intersection of the moment curve and the hyperplane where $\mu_4(t_1) = \mu_4(t_2) = \mu_4(t_3) = \mu_4(t_4) = 0$, corresponding to the Gale string $s = \mathbf{1111}00 \in G(4,6)$.

Example 2.11. As a counterexample, consider Figure 2.12. The points $t = t_4$ and $t' = t_6$ lie on the moment curve but are on opposite sides of the hyperplane H. The corresponding bitstring is $s = \mathbf{1}01\mathbf{1}0\mathbf{1}0$, which is not a Gale string; the violation of the Gale evenness condition correspond to the change of sign between the t_j's. An analogous case is shown in Figure 2.13; the corresponding bitstring is $s = \mathbf{111}0\mathbf{1}0$.

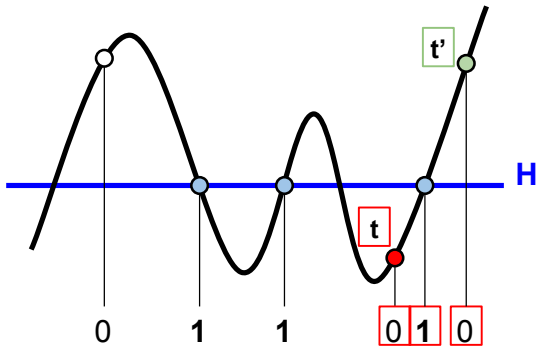


Figure 2.12 There is a change of sign between two t_j's; the bitstring $s = 0\mathbf{11}0\mathbf{1}0$ does not satisfy the Gale evenness condition, and the t_j's do not correspond to a facet of $C_3(6)$.

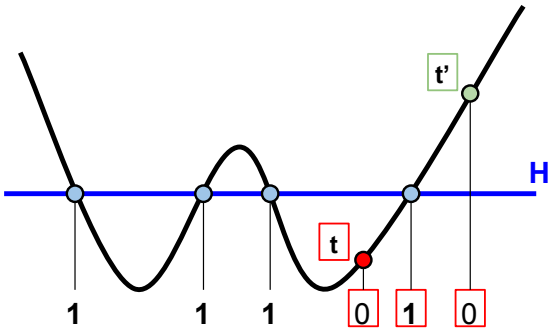


Figure 2.13 There is a change of sign between two t_j's; the bitstring $s = \mathbf{111}0\mathbf{1}0$ does not satisfy the Gale evenness condition, and the t_j's do not correspond to a facet of $C_4(6)$.

We now apply Theorem 11 to the study of bimatrix games. Proposition 2.2 states that 2-NASH can be reduced to ANOTHER COMPLETELY LABELED FACET. If the polytope Q in Theorem 5 is cyclic and we define a labeling for Gale strings such that a completely labeled Gale string corresponds to a completely labeled facet of Q, we can study unit vector games and their dual cyclic best response polytope as Gale strings. We say that $s \in G(d,n)$ is a *completely labeled Gale string* for some labeling function $l_s : [n] \to [d]$ if $\{l_s(i) \mid s(i) = \mathbf{1} \text{ for } i \in [n]\} = [d]$. Since $s \in G(d,n)$ has exacty d bits equal to $\mathbf{1}$, this means that for each $j \in [d]$ there is exactly one $i \in [n]$ such that $s(i) = \mathbf{1}$ and $l_s(i) = j$. Notice that it is not always possible to find a completely labeled Gale string.

Example 2.12. For $l_s = 121314$, there are no completely labeled Gale strings. The labels $l_s(i) = 2, 3, 4$ appear only once in l_s, so we must have $s(2) = s(4) = s(6) = 1$. We also must have $l_s(i) = 1$ for exactly one $i = 1, 3, 5$. The candidate strings are then $s = \mathbf{110101}$, $s' = 0\mathbf{11101}$, $s'' = 01\mathbf{0111}$; but none of these satisfies the Gale evenness condition.

A *Gale game* is a unit vector game (U, B) where $U = (e_{l(1)}, \ldots, e_{l(d)})$ for some labeling $l : [n] \to [d]$ and for which the dual of the best response polytope is a cyclic polytope $Q = \text{conv}\{e_1, \ldots, e_d, c_1, \ldots, c_n\}$. Theorem 5 gives the labeling (2.13) for the $d + n$ vertices of Q as

$$\begin{aligned} l_v(-e_i) &= i \qquad \text{for } i \in [m]; \\ l_v(c_j) &= l(j) \qquad \text{for } j \in [n]. \end{aligned}$$

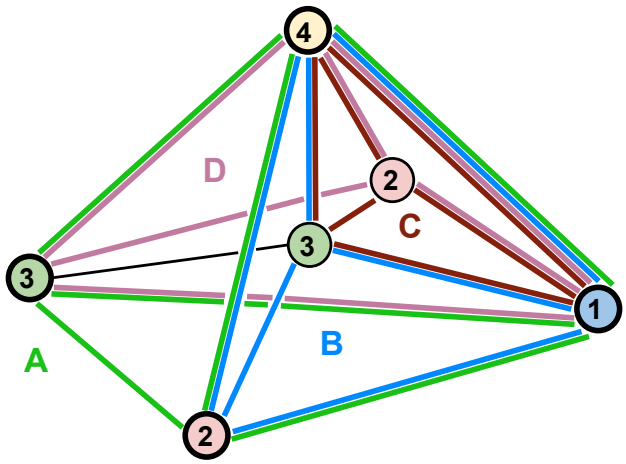
We define the labeling $l_s : [d + n] \to [d]$ of $G(d,n)$ as

$$\begin{aligned} l_s(i) &= i \qquad \text{for } i \in [d]; \\ l_s(d + j) &= l(j) \qquad \text{for } j \in [n]. \end{aligned} \tag{2.17}$$

Then the Gale strings $s \in G(d, d + n)$ that are completely labeled by l_s correspond exactly to facets of Q that are completely labeled by l_v, with the facet F_0 corresponding to the "trivial" completely labeled string $\mathbf{1}^d 0$.

From this point forward, we will assume that d is even; we will also assume that the labeling $l_s : [n] \to [d]$ satisfies $l_s(i) \neq l_s(i+1)$; this can be done without loss of generality, given the following consideration. Suppose that $l_s(i) = l_s(i+1)$ for some index i, and let s be a completely labeled Gale string for l_s; then only one of $s(i)$ and $s(i+1)$ can be equal to **1** (notice that it's possible that both are 0s), so $s(i)s(i+1)$ will never be a run of even length that "interferes" with the Gale Evenness Condition. Therefore, we can identify the indices i and $i+1$.

Example 2.13. Given the string of labels $l_s = 123432$, there are four associated completely labeled Gale strings in $G(4,6)$: $s_A = \mathbf{111}00$, $s_B = \mathbf{110}1\mathbf{1}0$, $s_C = \mathbf{1}00\mathbf{111}$ and $s_D = \mathbf{1}0\mathbf{11}0\mathbf{1}$. These correspond to the completely labeled facets for the labeling shown in Figure 2.14, left.



facet	**1 2 3 4 3 2**
A	**1 1 1 1** 0 0
B	**1 1** 0 **1 1** 0
C	**1** 0 0 **1 1 1**
D	**1** 0 **1 1** 0 **1**

Figure 2.14 The cyclic polytope $C_4(6)$; the labeling of the vertices correspond to the labeling of $G(4,6)$ given by $l_s = 123432$.

We can now define the problem Another Gale as in Table 2.3.

Another Gale	
input :	A labeling $l : [n] \to [d]$, where d is even and $d < n$. A Gale string $s \in G(d, n)$, completely labeled by l.
output:	A Gale string $s' \in G(d, n)$, completely labeled by l, such that $s' \neq s$.

Table 2.3 The problem Another Gale.

It takes polynomial time to translate the facets of the cyclic polytope $C_d(n)$ into the corresponding Gale strings in $G(d, n)$, following the proof of Theorem 11. Furthermore, defining the labeling l_s from the labeling l_v also takes polynomial time: for the labels $i \in [d]$ it is immediate, for the labels $d + j$, where $j \in [n]$, we have to check the $d \times n$ matrix U of the imitation game. Therefore, by Proposition 2.2, we have a reduction from Gale Nash to Another Gale.

Proposition 2.3. Gale Nash *is polynomial-time reducible to* Another Gale.

Chapter 3

Algorithmic and Complexity Results

In the previous chapter we have defined some problems of the form "find another completely labeled..." for vertices, facets and Gale strings. In this chapter we will finally study the complexity of these problems. The first section will present different versions of the standard algorithm introduced by Lemke and Howson [13]. We will first apply it to the problem Another Completely Labeled Vertex, in particular as a tool to find a solution for 2-Nash, its original motivation; we will then touch on its dual version for the problem Another Completely Labeled Facet; finally, we will describe the Lemke-Howson for Gale Algorithm, which returns a solution to Another Gale. We will follow each one of these algorithms with a very straightforward proof of the **PPA** complexity of Another Completely Labeled Vertex and Another Completely Labeled Facet; in the case of Another Galewe will give a result of **PPAD** complexity. We will close the section with an example of labeling, due to Morris [17], for which the Lemke-Howson for Gale algorithm presents exponential running time; this is the labeling that Savani and von Stengel [22] [23] have used to build Gale games that are "hard to solve".

The second section presents our original result: a polynomial time algorithm for ANOTHER GALE, therefore proving that GALE NASH is in **FP**. Unless **PPAD**=**P**, this goes in the opposite direction of our first conjecture of PPAD-completeness suggested by the "hard to solve" games. Our proof relies on a theorem by Edmonds [6] that gives a polyomial-time algorithm to find a perfect matching of a graph, or deciding that it is not possible to find one. The key of the proof is the construction of a graph from any string of labels; the perfect matchings of the graph will correspond to the completely labeled Gale strings for the labels. We will first prove the **FP** complexity of finding one of these completely labeled Gale strings; we will then extend the proof to find the second string required by ANOTHER GALE.

3.1 Proofs by Parity

In [15], Megiddo and Papadimitriou introduced the classes **FNP**, *function non-deterministic polynomial*, and **TFNP**, *total function non-deterministic polynomial*. The former is defined as the class of binary relations $R(x, y)$ such that there is a polynomial-time algorithm that decides $R(x, y)$ for x, y such that $|y| \leq p(|x|)$, where p is a polynomial. The latter is the class of all such problems for which y is guaranteed to exist. **FNP** and **TFNP** can be seen as the equivalent of **NP** for (respectively) function and total function problems. Also in [15], Megiddo and Papadimitriou proved that **TFNP** is a *semantic* class, that is, a class without complete problems, unless $\mathbf{NP} = \mathbf{co} - \mathbf{NP}$, which is to this day an open problem. To circumvent this limitation, Papadimitriou [21] focused on the problems for which the existence of a solution is proved by a specific argument, introducing the classes **PPA** (*Proof by Parity Argument*) and **PPAD** (*Proof by Parity Argument, Directed version*).

The existence of a solution for a problem in **PPA** can be proved using the argument "in any undirected graph with one odd-degree node there must be another odd-degree node." It is interesting to notice that **PPA**-complete

problems are yet to be found. Problems in **PPAD**, analogously, are guaranteed to have a solution by a proof employing the argument "in any directed graph in which all vertices have indegree and outdegree at most one where there is a *source* (a node with indegree zero) there must be a *sink* (a node with outdegree zero)." Formally, we can define **PPAD** as the class of problems reducible to the problem End Of The Line in Table 3.1. This is the definition given in Daskalakis, Goldberg and Papadimitriou [5]; the original definition in Papadimitriou [21] is given in terms of Turing machines. A *circuit* with n *input bits* and m *output bits* is a function $C : \{0,1\}^n \to \{0,1\}^m$.

End Of The Line	
input :	Two circuits S and P with n input bits and n output bits such that $P(0^n) = 0^n \neq S(0^n)$.
output:	An input $x \in \{0,1\}^n$ such that $P(S(x) \neq x)$ or $S(P(x)) \neq x \neq 0^n$

Table 3.1 The problem End Of The Line.

The problems in **PPAD** can be seen as a circuit S ("successor"), and a circuit P ("predecessor") that are used to build a directed graph with an edge (x, y) if and only if $S(x) = y$ and $P(y) = x$; furthermore, a *standard source* 0^n is given, guaranteeing the existence of the output, which consists in either a sink or a non-standard source. Figure 3.1 presents an example of a graph implicit in a **PPAD** problem; a graph for a **PPA** problem is analogous, but it is undirected and instead of sources and sinks there are generic endpoints.

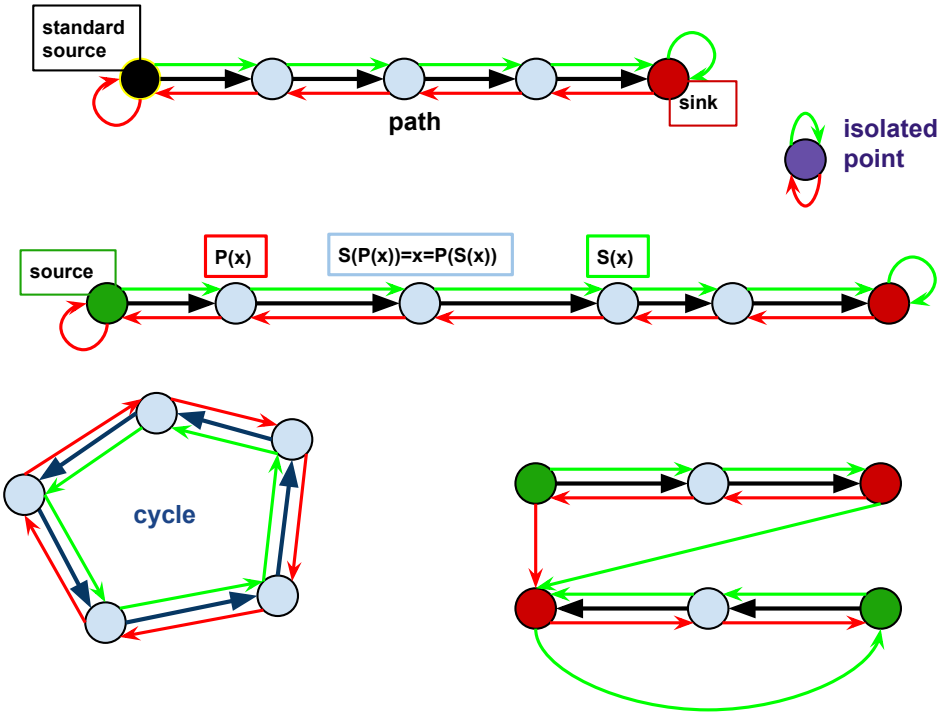


Figure 3.1 The graph in a **PPAD** problem (paths in black, cycles in blue, isolated points in purple). The input is given by the circuits S (in green) and P (in red) and the standard source (the black node). The output can be either a sink (a red node) or a nonstandard source (a green node).

By theorem 1, the problem n-NASH of Table 1.3 is a total function problem. Megiddo and Papadimitriou ([15]) proved that n-NASH is in **TFNP**. Daskalakis, Goldberg and Papadimitriou [5] and Chen and Deng [4] have later proven its **PPAD**-completeness, the former for $n \geq 3$ and the latter for $n \geq 2$. A small amendment of the proof in [5] can be found in Casetti [2].

Theorem 7. (Daskalakis, Goldberg and Papadimitriou [5]; Chen and Deng [4]) *For $n \geq 2$, the problem n-NASH is* **PPAD***-complete.*

We will see more problems in **PPA** and **PPAD** in chapter 3; in fact, our main result can be seen as a negative result on the **PPAD** complexity of a case of 2-NASH.

3.2 The Lemke-Howson Algorithm

Let P be a simple d-polytope with n facets. We *pivot on the vertices* of P by moving from a vertex x to another vertex y connected to x by an edge. Note that, since P is simple, there are exactly d possible choices for y. Analogously, we *pivot on the facets* of a simplicial polytope Q in dimension d by moving from a facet F to a facet G that share all vertices but one; since Q is simplicial, there are d possible choices for G.

Suppose now that there is a labeling $l_f : [n] \to [d]$ of the facets of the simple polytope P. If we pivot from a vertex x to a vertex x' we "leave behind" a facet F with label k; so, if x has labels $(l_1, \ldots, k, \ldots, l_d)$, then x' has labels $(l_1, \ldots, h, \ldots, l_d)$, where h is the label of the facet F' that does not have x as its vertex. We call this *dropping label k and picking up label h*, or *pivoting on label k*; see Figure 3.2. Analogously, if there is a labeling $l_v : [n] \to [d]$ of the vertices of the simplicial poytope Q and we pivot from a facet F with labels $(l_1, \ldots, k, \ldots, l_d)$ to a facet F' with labels $(l_1, \ldots, h, \ldots, l_d)$, we say that we *drop label k and pick up label h*, or that we *pivot on label k*; see Figure 3.3.

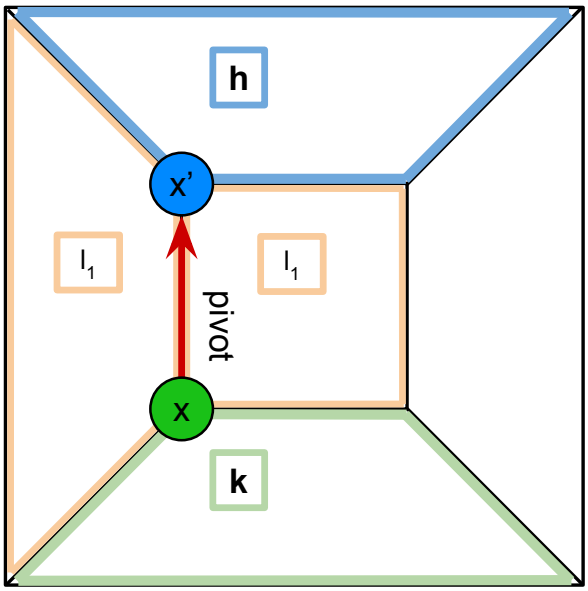


Figure 3.2 A pivot on the vertices of the cube.

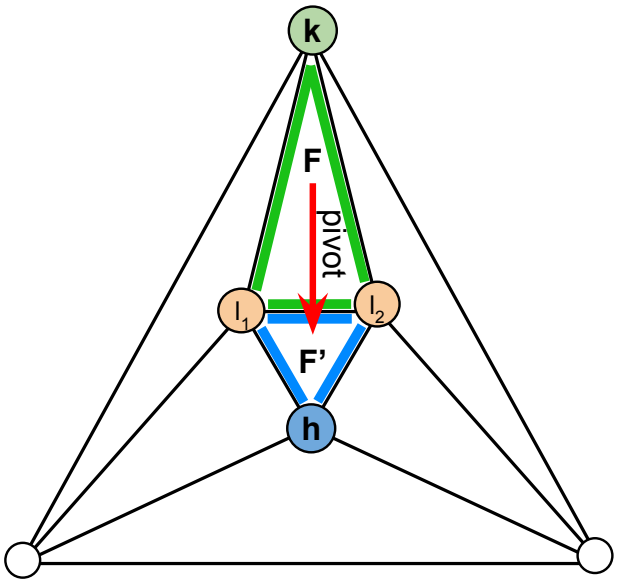


Figure 3.3 A pivot on the facets of the octahedron.

Let $m, n \in \mathbb{N}$ with $m \leq n$; consider a set X and a labeling $l : X \to [m]$. The n-uple $x = (x_1, \ldots, x_n) \in X^n$ is *almost completely labeled* if there is exactly one $k \in [m]$ such that $\{j \in [m] \mid x_i = j \text{ for some } i \in [n]\} = [m] \setminus \{k\}$. That is, an n-uple x is almost completely labeled if all labels appear once in x except for the *missing label* $k \in [m]$ and the *duplicate label* $h \in [m]$ that appears twice. It's easy to see that if we pivot from an almost completely labeled facet (or vertex) on the duplicate label, or from a completely labeled facet (or vertex) on any label, we reach either an almost completely labeled or a completely labeled facet (or vertex).

Algorithm 1: Lemke-Howson

input : A simple d-polytope P with n facets. A labeling $l_f : [n] \to [d]$ of the facets of P. A vertex x_0 of P, completely labeled for l.

output: A completely labeled vertex $x \neq x_0$ of P.

```
choose any label k ∈ [d]
pivot on label k from x_0 to x
while x is not completely labeled do
    pivot on the duplicate label h from x to x' ≠ x_0
    set x_0 = x, x = x'
return x
```

The classic Lemke-Howson Algorithm 1, first introduced by Lemke and Howson [13], employs pivoting on the vertices of a simple polytope, moving through a succession of almost completely labeled vertices with missing label k; these vertices can be seen as steps of a path, called *Lemke path*. We will use Lemke paths to prove some fundamental properties of both the Lemke-Howson Algorithm and the problem Another Completely Labeled Vertex.

Proposition 3.1. *The Lemke-Howson Algorithm 1 returns a solution to the* **PPA** *problem* Another Completely Labeled Vertex.

Furthermore, the number of completely labeled vertices in a simple polytope

with labeled facets is even.

Proof. We first show that the Lemke-Howson Algorithm works. Let x' be an almost completely labeled vertex of the Lemke path with duplicate label h. There are only two facets with duplicate label h that contain x; since P is simple, each one of which corresponds to exactly an edge. One of these edges has been traversed to get to x', the other edge will be traversed to leave x' in the next step. Therefore, there are no "loops" where a vertex is visited more than once; the Lemke paths are *simple*.

The parity is proven by the following argument: each Lemke path is uniquely determined by its missing label and its starting point, so the Lemke path from the endpoint with the same missing label will lead back to the starting point. Since the endpoint and the starting point are different, the Lemke paths must connect an even number of points.

Finally, for each label $k \in [d]$ chosen in line 1 of Algorithm 1, the Lemke paths are disjoint paths connecting all the completely labeled vertices of P, with a standard endpoint x_0. The problem Another Completely Labeled Vertex correspond to finding a non-standard endpoints of this graph, which is a **PPA** problem. □

Applying the parity result of Proposition 3.1 to the case of a bimatrix game (not necessarily a unit vector game), and remembering that the point $(\mathbf{0}, \mathbf{0})$ corresponds to the "artificial" equilibrium, we have the following result, due to Lemke and Howson [13].

Theorem 8. (Lemke-Howson [13]) *Every non-degenerate bimatrix game has an odd number of Nash equilibria.*

There are two ways of using the Lemke-Howson Algorithm to find a Nash equilibrium of a bimatrix game (A, B). The first one is to "symmetrize" the game as in Proposition 2.1. Let $S = \{z \in \mathbb{R}^{m+n} \mid z \geq \mathbf{0},\ Cz \leq \mathbf{1}\}$ be the polytope associated to the game $(C, C^\top)$, where $C = \left(\begin{smallmatrix} 0 & A \\ B^\top & 0 \end{smallmatrix}\right)$. The facets

of C correspond to $2(m+n)$ inequalities; we label both the i-th and the $(m+n+i)$-th inequality as $i \in [m+n]$ and we apply the Lemke-Howson Algorithm starting from the vertex $\mathbf{0}$. This returns a Nash equilibrium (z, z) of C, which corresponds to a Nash equilibrium $(x, y) = z$ of (A, B). We can also follow the "traditional" version of the Lemke-Howson Algorithm alternating a move on the best response polytopes P and a move on the best response polytope Q of (2.4). Since the polytopes P and Q are in $\mathbb{R}^m$ and $\mathbb{R}^n$, whereas S is a polytope in $\mathbb{R}^{m+n}$, this second version is much easier to visualize.

Example 3.1. Consider the 3×3 game (A, B) of Example 2.1.

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}.$$

The best response polytopes can be represented as the best response regions of Figure 2.1 extended to the origin $\mathbf{0}$, as in Figure 3.4. The path starts in $(\mathbf{0}, \mathbf{0})$; we drop the label 2 and we move on the polytope P. The label 6 is duplicate; so we drop it and we make the next move on the polytope Q, and so on until we reach the point x, that is the only Nash equilibrium of (A, B).

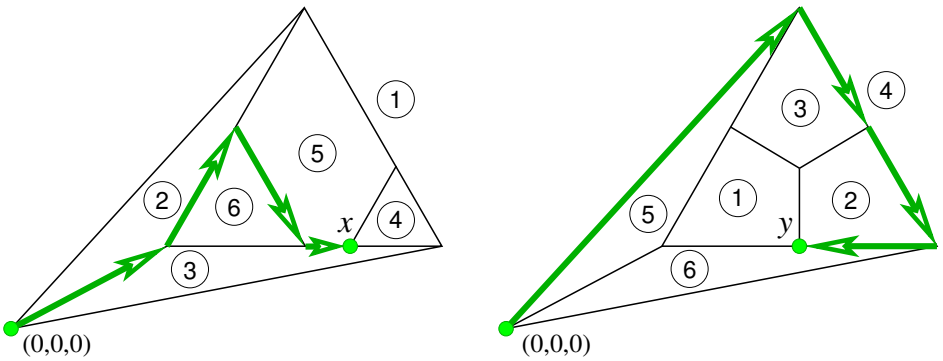


Figure 3.4 Lemke path for missing label 2 on the best response polytopes P (left) and Q (right) of game (2.2).

It is possible to have an equilibrium that cannot be reached applying the Lemke-Howson Algorithm from the artificial equilibrium, or even from the endpoint of a Lemke path from the artificial equilibrium. This is can be seen in the next example, due to Wilson, given in Shapley [24]. Notice that, by the parity result in Proposition 3.1, there must be at least another equilibrium that is "disconnected" from the artificial one in this way.

Example 3.2. Consider the symmetric game $(C, C^\top)$ with

$$C = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 2 & 0 \\ 3 & 0 & 1 \end{pmatrix}. \tag{3.1}$$

There are three equilibria of $(C, C^\top)$, all of them symmetric, at (x_i, x_i) with $x_1 = (0, 0, 1)$, $x_2 = (1/6, 1/3, 1/2)$ and $x_3 = (1/3, 2/3, 0)$. All Lemke paths from the artificial equilibrium $(0, 0)$ end at (x_1, x_1), and consequently all other Lemke paths connect (x_2, x_2) and (x_3, x_3); see Figure 3.5.

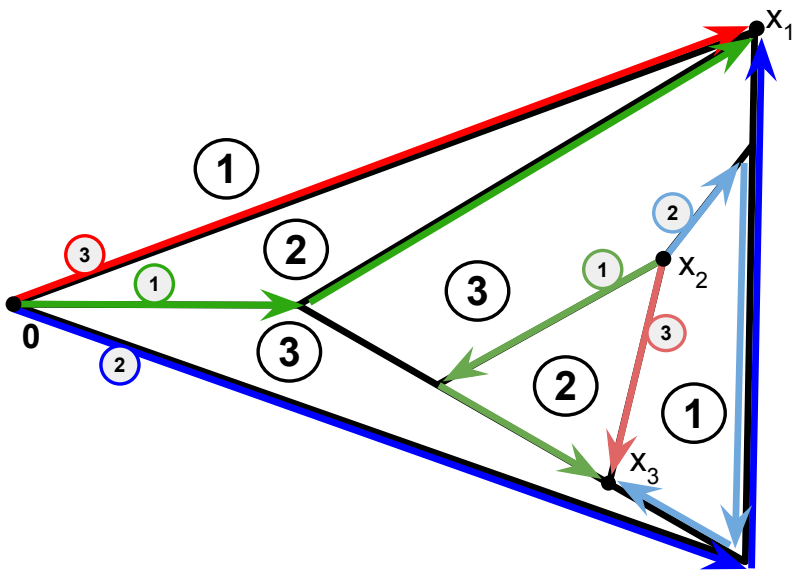


Figure 3.5 Lemke paths on the best response polytope of game (3.1).

The dual version of the Lemke-Howson Algorithm 1 and of proposition 3.1 is quite straightforward.

Algorithm 2: Dual Lemke-Howson

input : A simplicial m-polytope Q with n vertices. A labeling $l_v : [n] \to [d]$ of the vertices of P. A vertex F_0 of Q, completely labeled for l.

output: A completely labeled facet $F \neq F_0$ of Q.

```
choose any label k ∈ [d]
pivot on label k from F_0 to F
while x is not completely labeled do
    pivot on the duplicate label h from F to F' ≠ x_0
    set F_0 = x, F = F'
return x
```

Proposition 3.2. *The Dual Lemke-Howson Algorithm 2 returns a solution to the* **PPA** *problem* Another Completely Labeled Facet.

Furthermore, the number of completely labeled facets in a simplicial polytope with labeled vertices is even.

By Theorem 4 and Theorem 5, in the case of unit vector games it is enough to apply the Lemke-Howson Algorithm to the polytope $P^l = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}$ in (2.9), or the Dual Lemke-Howson Algorithm to the polytope $Q = \text{conv}(\{e_1, \ldots, e_m\}) \cup \{c_1, \ldots, c_n\})$ in (2.11). The following theorem by Savani and von Stengel [23] guarantees that not only this yield a Nash equilibrium, but no potential solutions are "lost" considering the polytope P^l with m labels instead of the product of polytopes $P \times Q$ with $m + n$ labels; an analogous result holds for the dual case.

Theorem 9. *Let (U, B) be a unit vector game, with $U = (e_{l(1)} \cdots e_{l(n)})$ for a labeling $l : [n] \to [m]$. Let $P = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}$ and $Q = \{y \in \mathbb{R}^n \mid y \geq \mathbf{0},\ Ay \leq \mathbf{1}\}$ as in (2.4); let $P^l = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}$ as in (2.9). Then for the missing label $k \in [m]$ the Lemke path on $P \times Q$ projects to a path on P that corresponds to the Lemke path on P^l for the missing label k; for the missing label $k = m + j$, where $j \in [n]$, the Lemke path on $P \times Q$ projects to a path on Q that corresponds to the Lemke path on P^l for the missing label $l(j)$.*

We finally focus on the case of unit vector games where the simplicial polytope Q is cyclic, which we can study from the point of view of Gale strings. We will consider $s \in G(d, n)$, with d even, as "loops". Let $s(i) = 1$ for an index $i \in [n]$; then, by Gale evenness condition, there is an odd run of **1** 's either on the left or on the right of position i in s; let j be the first index after this run. A *pivot on s* is given by setting $s(i) = 0$ and $s(j) = 1$. If there is a labeling $l_s : [n] \to [m]$, we say that we *pivot on label $l_s(i)$*, *dropping label $l_s(i)$* and *picking up label $l_s(j)$*. The *Lemke Howson for Gale Algorithm* is given as Algorithm 3.

Algorithm 3: Lemke-Howson for Gale

input : A labeling $l_s : [n] \to [d]$, where d is even, such that there is a completely labeled Gale string $s_0 \in G(d, n)$.

output: A completely labeled Gale string $s \in G(d, n)$ such that $s \neq s_0$.

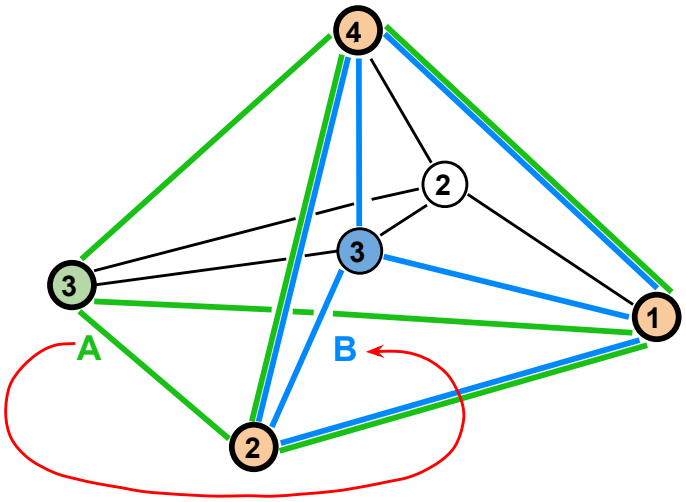
```
choose a label k ∈ [d]
pivot on label k from s_0 to s
while s is not completely labeled do
    pivot on the duplicate label h from s to s' ≠ s_0
    rename s_0 = s, s = s'
return s
```

The next example illustrates the correspondence between the Lemke-Howson Algorithm and the Lemke-Howson for Gale Algorithm.

Example 3.3. Figure 3.6 shows the cyclic polytope $C_4(6)$ with the labeling

$$\begin{aligned} l_v(i) &= i \qquad \text{for } i \in [4]; \\ l_v(5) &= 3; \\ l_v(6) &= 2. \end{aligned}$$

This corresponds to the labeling l_s for $G(4,6)$ given in Example 2.13, which gives the four completely labeled Gale strings $s_A = \mathbf{111}00$, $s_B = \mathbf{110}1\mathbf{1}0$, $s_C = 0\mathbf{0111}$ and $s_D = \mathbf{101}0\mathbf{1}$. These correspond to the facets A, B, C and D of $C_4(6)$. Pivoting from $s_A = \mathbf{111}00$ dropping label 3 returns $s_B = \mathbf{110}1\mathbf{1}0$, analogously, pivoting from facet A dropping label 3 returns facet B.



facet	**1 2 3 4 3 2**
A	**1 1** $\underline{1}$ **1** 0 0
B	**1 1** 0 **1** $\overline{1}$ 0
facet	**1 2 3 4 3 2**

Figure 3.6 The pivoting to $s_A = \mathbf{111}00$ to $s_B = \mathbf{110}1\mathbf{1}0$ in the Lemke-Howson for Gale Algorithm corresponds to the pivoting from facet A to facet B in the Dual Lemke-Howson Algorithm.

We will now show a construction that we will allow us to extend the analogous of Proposition 3.1 for the Lemke-Howson for Gale Algorithm to prove the **PPAD** complexity of ANOTHER GALE. A *permutation* of the elements of an ordered set S is a sequence without repetition of elements of S; a *transposition* is a permutation of exactly two elements of S. The *sign of a permutation* σ is $\text{sign}(\sigma) = (-1)^m$, where m is the number of transpositions needed to get $\sigma_0 = 1 \dots n$ from σ. It is immediate to see that any two permutations that differ in only one transposition have opposite sign. The *sign of a completely labeled Gale string* $s \in G(d, n)$ is defined as follows: let $l : [n] \to [d]$ be the relative labeling of $G(d, n)$, and let l_0 be the string of labels $l(i)$ such that $s(i) = 1$ and that two labels corresponding to a run in l are adjacent in l_0. Then we define $\text{sign}(s) = \text{sign}(l_0)$. Notice that if $l(i) = i$ for $i \in [d]$ then the sign of the completely labeled Gale string $1^d 0^{(n-d)}$ is always positive. The *sign of an almost completely labeled Gale string* $s \in G(d, n)$ with missing label k and duplicate label h is defined on two different strings. Let i_1 be the index of h reached by the last pivot (the "new" position of the **1**) and let i_2 be the index of h such that $s(i_2) = 1$ before the last pivot (the "old" position of the **1**). Let l_1 be the string obtained as l_0 substituting k to h at index i_1, and let l_2 be the string obtained as l_0 substituting k to h at index i_2. Notice that $\text{sign}(l_1) = -\text{sign}(l_2)$, since they can be obtained from each other applying the transposition (kh).

Consider now the steps of the Lemke paths in the Lemke-Howson for Gale Algorithm; we assume that $\text{sign}(s_0) = +1$ without loss of generality, since the negative case is the same just with opposite signs. If the first pivot returns another completely labeled Gale string s, this must have negative sign because it has been obtained "jumping" over an odd number of **1**'s. For the same reason, if the pivoting returns an almost completely labeled Gale string, we have that $\text{sign}(l_1) = -1$, therefore $\text{sign}(l_2) = +1$. The next pivoting step drops the label h from index i_2, so again we change sign. Running the Lemke-Howson

for Gale Algorithm will therefore result in the sign of the completely labeled Gale strings swinging as in Table 3.2. Notice that all this construction can be done in polynomial time; orienting all Lemke paths from positive to negative reduces the problem Another Gale to End Of The Line.

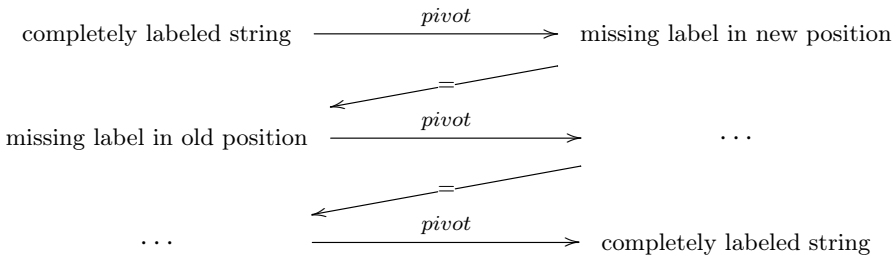


Table 3.2 Sign switching on the Lemke-Howson for Gale Algorithm.

Proposition 3.3. *The Lemke-Howson for Gale Algorithm 3 returns a solution to the* **PPAD** *problem* Another Gale.

Furthermore, the number of completely labeled Gale strings $s \in G(d, n)$, where d is even, is even.

A result similar to Proposition 3.3, but more general, is given in Shapley [24]: two Nash equilibria at the ends of a Lemke path have opposite *index*, which is analogous to the sign but defined using determinants on the payoff matrices for the equilibrium support; the artificial equilibrium is assigned index $+1$. This is used to prove that a nondegenerate game with n Nash equilibria with index $+1$ has $n + 1$ Nash equilibria with index -1.

Example 3.4. Let $l_s = 123432$; consider the Lemke path from the completely labeled Gale string $s = \mathbf{111}00$ dropping label 1. Figure 3.7 shows the graph of Table 3.2. Notice that $\text{sign}(\mathbf{101101}) = \text{sign}(l(6)l(1)l(3)l(4)) = \text{sign}((2134))$, since $s(6) = s(1) = 1$ and therefore the indices 6 and 1 are consecutive in the same run.

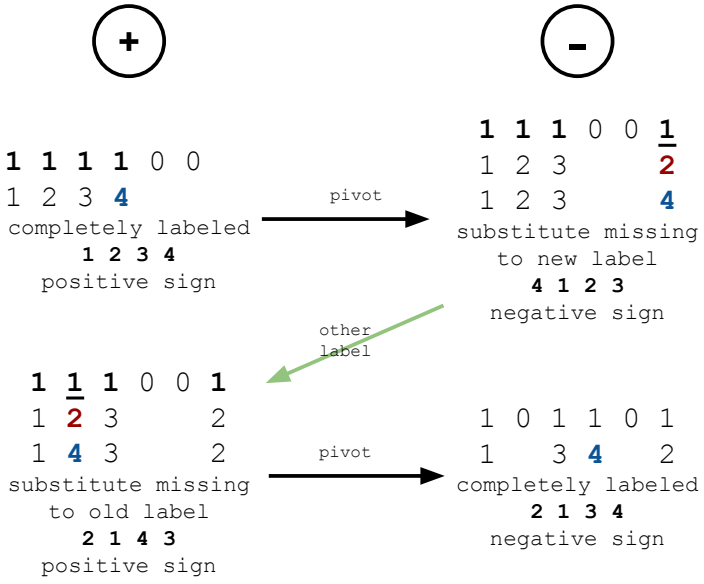


Figure 3.7 Pivoting with sign on 123432.

An example of labeling $l : [2d] \to [d]$ where the length of the Lemke paths on the cyclic polytope $C_d(2d)$, therefore on $G(d, 2d)$, grows exponentially in d, regardless of the label that is dropped, is given by Morris [17]. This labeling is given for d both even and odd as follows:

$$
\begin{aligned}
l(k) &= k && \text{for } k \in [d];\\
l(d+1) &= d;\\
l(d+k) &= d-k && \text{for } 2 \le k < d,\ k \text{ even};\\
l(d+k) &= d-k+2 && \text{for } 2 \le k \le d,\ k \text{ odd};\\
l(2d) &= 1 && \text{for } k \text{ even}.
\end{aligned}
$$

Avoiding repetitions and restricting to the case of d even, as we assumed, the Morris' labeling is simplified to $l : [2d-2] \to [d]$ as follows:

$$\begin{aligned} l(k) &= k && \text{for } k \in [d]; \\ l(d+k) &= d-k+1 && \text{for } k \in [d],\ k \text{ even}\ ; \\ l(d+k) &= d-k-1 && \text{for } k \in [d],\ k \text{ odd}\ . \end{aligned}$$

Example 3.5. Consider the labeling $l = 1234564523$ for $G(6,10)$. The only two completely labeled Gale string are $s = \mathbf{11111}00$ and $s' = \mathbf{1}0000\mathbf{11111}$. Figure 3.8 shows the Lemke path for label 1.

1	**2**	**3**	**4**	**5**	**6**	**4**	**5**	**2**	**3**
$\underline{\mathbf{1}}$	**1**	**1**	**1**	**1**	**1**	0	0	0	0
0	**1**	**1**	$\underline{\mathbf{1}}$	**1**	**1**	$\overline{\mathbf{1}}$	0	0	0
0	**1**	**1**	0	$\underline{\mathbf{1}}$	**1**	**1**	$\overline{\mathbf{1}}$	0	0
0	$\underline{\mathbf{1}}$	**1**	0	0	**1**	**1**	**1**	$\overline{\mathbf{1}}$	0
0	0	**1**	$\overline{\mathbf{1}}$	0	**1**	$\underline{\mathbf{1}}$	**1**	**1**	0
0	0	**1**	**1**	$\overline{\mathbf{1}}$	**1**	0	$\underline{\mathbf{1}}$	**1**	0
0	0	$\underline{\mathbf{1}}$	**1**	**1**	**1**	0	0	**1**	$\overline{\mathbf{1}}$
0	0	0	$\underline{\mathbf{1}}$	**1**	**1**	$\overline{\mathbf{1}}$	0	**1**	**1**
0	0	0	0	$\underline{\mathbf{1}}$	**1**	**1**	$\overline{\mathbf{1}}$	**1**	**1**
$\overline{\mathbf{1}}$	0	0	0	0	**1**	**1**	**1**	**1**	**1**
1	**2**	**3**	**4**	**5**	**6**	**4**	**5**	**2**	**3**

Figure 3.8 The Morris path for $C(6,10)$.

Savani and von Stengel [22] [23] relied on Morris' example to build different "hard to solve" games; these results give a strong motivation to study the compelxity of Another Gale. Our main result, in the next section, will give a **FP** algorithm that circumvents the problem of any exponential running time of the Lemke-Howson for Gale Algorithm.

3.3 The Complexity of Another Gale

A *matching* of a graph $G = (V, E)$ is a set $M \subseteq E$ such that every vertex $v \in V$ is the endpoint of at most one edge $m \in M$. A *perfect matching* is a matching such that there is an edge $m \in M$ incident to every vertex $v \in V$. Edmonds [6] proved the **FP** complexity of finding a perfect matching. The result can be easily extended to multigraphs.

Perfect Matching	
input	: A multigraph $G = (V, E)$.
output:	A perfect matching for G, or No if there is no possible perfect matching for G.

Table 3.3 The problem Perfect Matching.

Theorem 10. (Edmonds [6]) *It takes polynomial time to find the perfect matching of a graph or decide that no such matching exists.*

Proposition 3.4. Perfect Matching *is in* **PF**.

To prove our main result on Another Gale, we will prove that the accessory problem Gale can be solved in polynomial time.

Gale	
input	: A labeling $l : [n] \to [d]$, where d is even and $d < n$.
output:	A Gale string $s \in G(d, n)$ that is completely labeled by l

Table 3.4 The problem Gale.

Theorem 11. Gale *is in* **FP**.

Proof. Consider the multigraph $G = (V, E)$ with $V = [d]$, so that the vertices of G correspond to the labels $l(i) \in [d]$, and $E = \{(l(i), l(i+1)) \text{ for } i \in [n]\}$,

so that there is an edge between two vertices if and only if the corresponding labels are next to each other at some index i.

Let $s \in G(d, n)$ be a completely labeled Gale string. By the Gale evenness condition, every run of length k in s corresponds uniquely to $k/2$ pairs of indices $(i, i+1)$ with $s(i) = s(i+1) = 1$. Since s is completely labeled, all labels $l(i) \in [d]$ occur at exactly one of these indices and the edges $(l(i), l(i+1))$ form a perfect matching of G.

Conversely, let M be a perfect matching for $G = (V, E)$, defined as above. Let s be a bitstring with $s(i) = s(i+1)$ for every $(l(i), l(i+1)) \in M$ and $s(i) = 0$ otherwise. Since M is a matching, all the $(l(i), l(i+1)) \in M$ are disjoint, so every run of s is of even length; therefore s satisfies the Gale evenness condition. Furthermore, since M is perfect, s is completely labeled.

This proves that GALE reduces to PERFECT MATCHING in polynomial time; therefore, by Proposition 3.4, Algorithm 4 takes polynomial time to find a solution of an instance of GALE or deciding that no such solution exists. □

Algorithm 4: Completely Labeled Gale String Finder

input : A labeling $l : [n] \to [d]$.

output: A Gale string $s \in G(d, n)$ completely labeled by l.

```
1  Set G = (V, E), with V = [d] and E = ∅
2  Set s = 0^n
3  for i ∈ [n − 1] do
4      E = E ∪ {(l(i), l(i + 1))}
5  E = E ∪ {(l(n), l(1))}
6  Run the Edmonds' Algorithm on G
7  if The Edmonds' Algorithm returns a perfect matching M of G then
8      for (l(i), l(j)) ∈ M do
9          s(i) = s(j) = 1
10     Return s
11 else
12     Return No
```

We give an example of the construction, using the Morris' labeling of Example 3.5.

Example 3.6. Figure 3.9 shows the graph for the Morris labeling $l_s = 1234564523$, and its two matchings $M = \{e_1, e_3, e_5\}$ and $M' = \{e_8, e_6, e_{10}\}$. These, in turn, correspond to the completely labeled Gale strings $s = \mathbf{111111}0000$ and $s = 1000\mathbf{11111}$.

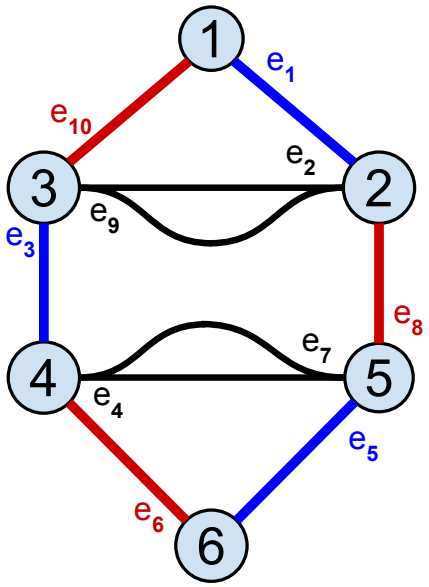


Figure 3.9 Two perfect matchings for the Morris labeling $l = 1234564523$.

Notice that the existence of a completely labeled Gale string is not guaranteed.

Example 3.7. Consider the labeling $l_s = 121314$. Figure 3.10 shows the corresponding graph, that has no possible perfect matching. Analogously, as we have seen in Example 2.12, there isn't any possible completely labeled Gale string for l_s.

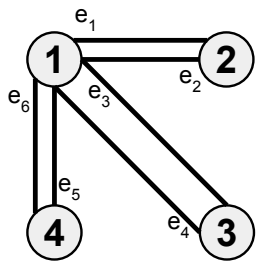


Figure 3.10 The graph for the labeling $l = 121314$

We finally extend the proof of Theorem 11 to the complexity of ANOTHER GALE.

Theorem 12. ANOTHER GALE *is in* **FP**.

Proof. Let $G = (V, E)$ be the graph corresponding to the labeling $l : [n] \to [d]$ as in the proof of Theorem 11 and let M_0 be the perfect matching of G corresponding to the completely labeled Gale string $s_0 \in G(d, n)$.

Theorem 3.3 guarantees the existence of another completely labeled Gale string $s \neq s_0$, therefore of the corresponding perfect matching $M \neq M_0$. Since $M \neq M_0$, there is at least one edge $e \in M_0$ such that $e \notin M$. Consider the $d/2$ graphs $G_i = (V, E_i)$, where $E_i = E \setminus \{e_i\}$ for $e_i \in M_0$. Since $V(G) = V(G_i)$ and $E(G_i) \subset E(G)$, if G_i has a perfect matching M_i, then this is a perfect matching for G as well. Since $e_i \notin E_i$, then $e_i \notin M_i$, and $M_i \neq M$. Since the number of G_i is exactly $d/2$, by Proposition 10 it takes at most polynomial time to find a perfect matching $M \neq M_0$ in one of the G_i's, and therefore the corresponding completely labeled Gale string $s \neq s_0$, applying Algorithm 5. □

Algorithm 5: Another Gale Finder

input : A labeling $l : [n] \to [d]$ and a Gale string $s_0 \in G(d, n)$ completely labeled by l.

output: A Gale string $s \in G(d, n)$ completely labeled by l, such that $s \neq s_0$.

Set $G = (V, E)$, with $V = [d]$ and $E = M_0 = \varnothing$
Set $s = 0^n$
for $i \in [n-1]$ **do**
    $E = E \cup \{(l(i), l(i+1))\}$
    if $s_0(i) = s_0(i+1) = 1$ **then**
        $M_0 = M_0 \cup \{(l(i), l(i+1))\}$
$E = E \cup \{(l(n), l(1))\}$
if $s_0(n) = s_0(1) = 1$ *and* $(s_0(1), s_0(2)) \notin M_0$ **then**
    $M_0 = M_0 \cup \{(l(n), l(1))\}$
for $m \in M_0$ **do**
    $E_m = E \setminus \{m\}$
    Run the Edmonds' Algorithm on $G_m = (V, E_m)$
    if *The Edmonds' Algorithm returns a perfect matching* M_m *of* G_m **then**
        for $(l(i), l(j)) \in M_m$ **do**
            $s(i) = s(j) = \mathbf{1}$
        Return s

We give two examples of the construction of theorem 12.

Example 3.8. The labeling $l = 123432$ gives the graph G in Figure 3.11. Suppose that Edmonds' algorithm returns the matching $M_0 = \{e_1, e_3\}$ of Figure 3.11 left, corresponding to the completely labeled Gale string $s_0 = \mathbf{111}00$; a second perfect matching $M = \{e_3, e_6\}$, corresponding to the Gale string $s = \mathbf{101101}$, can be found in the graph $G_1 = (V(G), E(G) \setminus \{e_1\})$, see Figure 3.11 right.

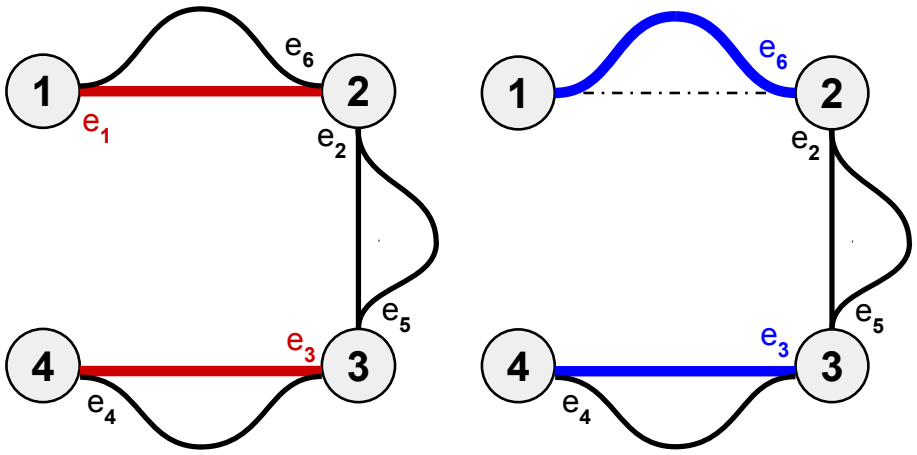


Figure 3.11 Left: The graph for the labeling $l = 123432$ with the perfect matchings corresponding to the completely labeled Gale strings $s_0 = \mathbf{111}00$. Right: The graph $G_1 = (V(G), E(G) \setminus \{e_1\})$ and the perfect matching corresponding to the Gale string $s = \mathbf{101101}$.

Example 3.9. Consider the Morris graph of Example 3.6; suppose that Edmonds' algorithm returns the perfect matching $M_0 = \{e_1, e_3, e_5\}$, as in Figure 3.12 left, corresponding to the completely labeled Gale string $s_0 = \mathbf{1111}10000$. We can then delete the edge e_1 to obtain the graph G_1, as in Figure 3.12 center. The graph G_1 has a perfect matching $M = \{e_6, e_8, e_{10}\}$, shown in Figure 3.12 right; this is also a perfect matching of G, corresponding to the only other completely labeled Gale string $s = 1000\mathbf{11111}$.

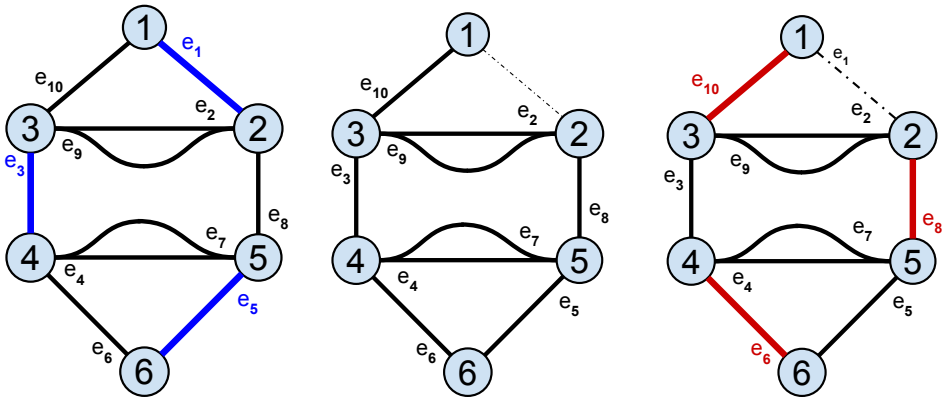


Figure 3.12 Left: The Morris graph $G = (V, E)$ with the matching $M = \{e_1, e_3, e_5\}$.
Centre: The graph $G_1 = (V(G), E(G) \setminus \{e_1\})$.
Right: The set $M' = \{e_6, e_8, e_{10}\}$ is a perfect matching of both G_1 and G.

Applying Theorem 2.3 to Theorem 12 we have our main result: Gale Nash is in **FP**.

Theorem 13. *Finding a Nash equilibrium of a Gale game takes polynomial time.*

Chapter 4

Further results

Algorithm 5 allows us to find a Nash equilibrium of a Gale game in polynomial time starting from another equilibrium (usually the artificial one), but it doesn't give any information on the relationship of these equilibria as endpoints of the Lemke-Howson algorithm. In particular, the issue of the sign of the equilibrium is left open. Following the construction in Proposition 3.3, we can reduce the problem "given a Nash equilibrium of a Gale game, find another one of opposite sign" to the problem Opposite Sign Gale of Table 4.1. Analogously, it is straightforward to see that these problems belongs to the class **PPADS**, see Daskalakis, Goldberg and Papadimitriou [5]; this is the equivalent of **PPAD** where the solution is restricted to sinks of the End Of The Line graph.

Opposite Sign Gale	
input :	A labeling $l : [n] \to [d]$ and a completely labeled Gale string $s_0 \in G(d, n)$.
output:	A completely labeled Gale string $s \in G(d, n)$ such that $\text{sign}(s) = -\text{sign}(s_0)$.

Table 4.1 The problem Opposite Sign Gale.

A first polynomial-time algorithm for Opposite Sign Gale of Table 4.1 was given by Merschen [16] in the case of labelings for which the Gale graph of Theorem 11 is planar; a general result has since been given by Végh and von Stengel [25]. The latter relies on the definition of a general framework to deal with pivoting algorithms, called *Complementary Pivoting with Direction Algorithm*, that is defined not only for perfect matchings of a Gale graph, but for the wider class of room partitions of Euler complexes.

A *d-dimensional Euler complex* (V, M) (in the following: *d-oik*) over a finite set of nodes V, as first introduced by Edmonds [7] and Edmonds and Sanità [**?**], is a collection of sets $M = \{R \mid R \subset V,\ |R| = d\}$, called *rooms*, such that any set of $d-1$ nodes, called *wall*, is contained in an even number of rooms; a *room partitioning* of (V, M) is $P \subset M$ such that each vertex of V is in exactly one room in P. Edmonds and Sanitá used an "exchange algorithm" and a parity argument to show that there is an even number of room partitions of (V, M). This can be applied to a family of two oiks (of possibly different dimension) corresponding to the best response polytopes of two players in a bimatrix game, with the room partitions corresponding to equilibria; the Lemke-Howson Algorithm is then a special case of the exchange algorithm. Another special case is an Euler graph: the edges are rooms, and perfect matchings are then room partitions. The connections between oiks and the topic of our study could appear trivial; but the issue of giving a sign to room partitions, unfortunately, is much more complex than that.

Example 4.1. Consider the octahedron with vertices labeled as in Figure 4.1; the endpoints of the Lemke paths in the Dual Lemke-Howson Algorithm 2 when dropping different labels are shown in Figure 4.2. Notice that this graph is bipartite: this corresponds to a parity argument with sign similar to Proposition 3.3.

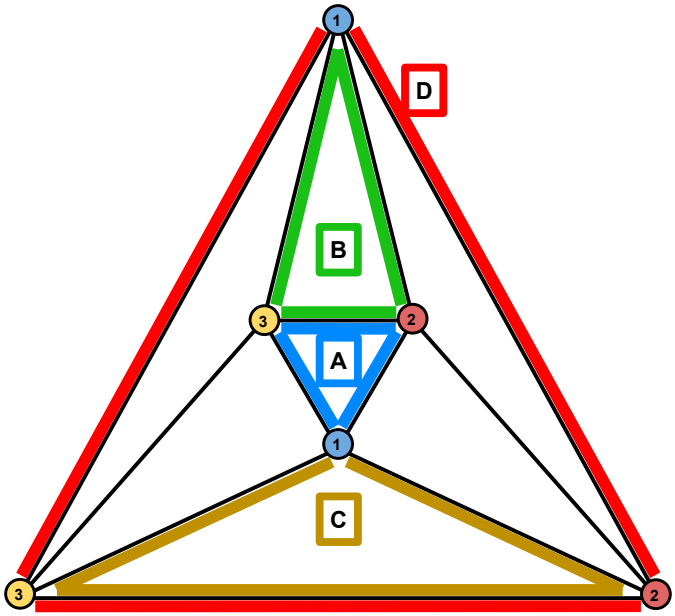


Figure 4.1 A labeled octahedron and its completely labeled facets.

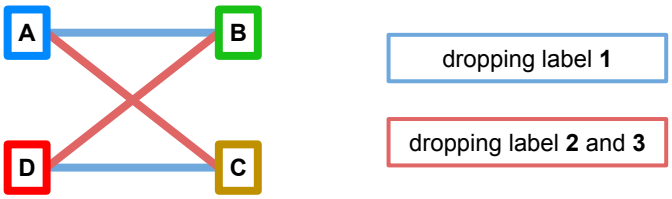


Figure 4.2 The endpoints of the Lemke paths on the octahedron in Figure 4.1.

On the other hand, consider the same octahedron from the point of view of room partitions, Figure 4.3. The exchange algorithm of Edmonds [7] from room partition B to room partition A dropping vertex $v = 1$ is shown in Figure 4.5. The graph describing its endpoints, analogous to Figure 4.2 is shown in Figure 4.4 and it is not bipartite. Giving a sign to the room partitions cannot be done simply through Edmonds' exchange algorithm.

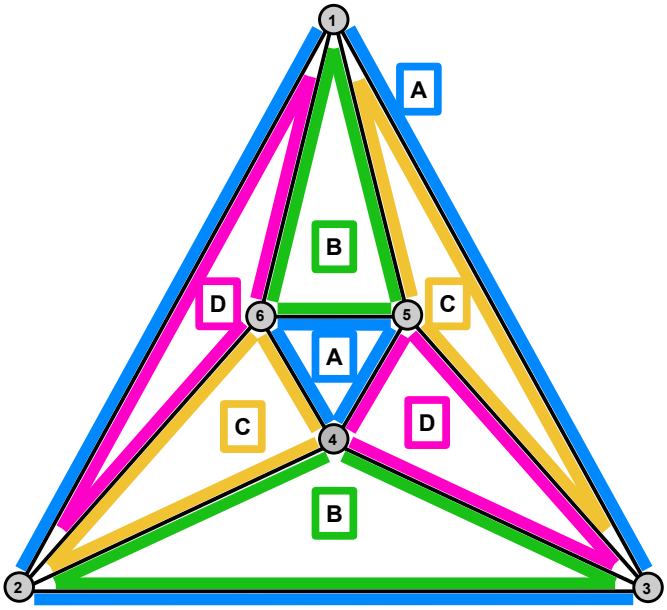


Figure 4.3 An octahedron and its room partitions.

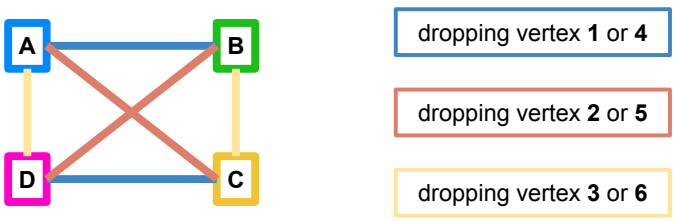


Figure 4.4 Endpoints of the Exchange Algorithm on the octahedron in Figure 4.3.

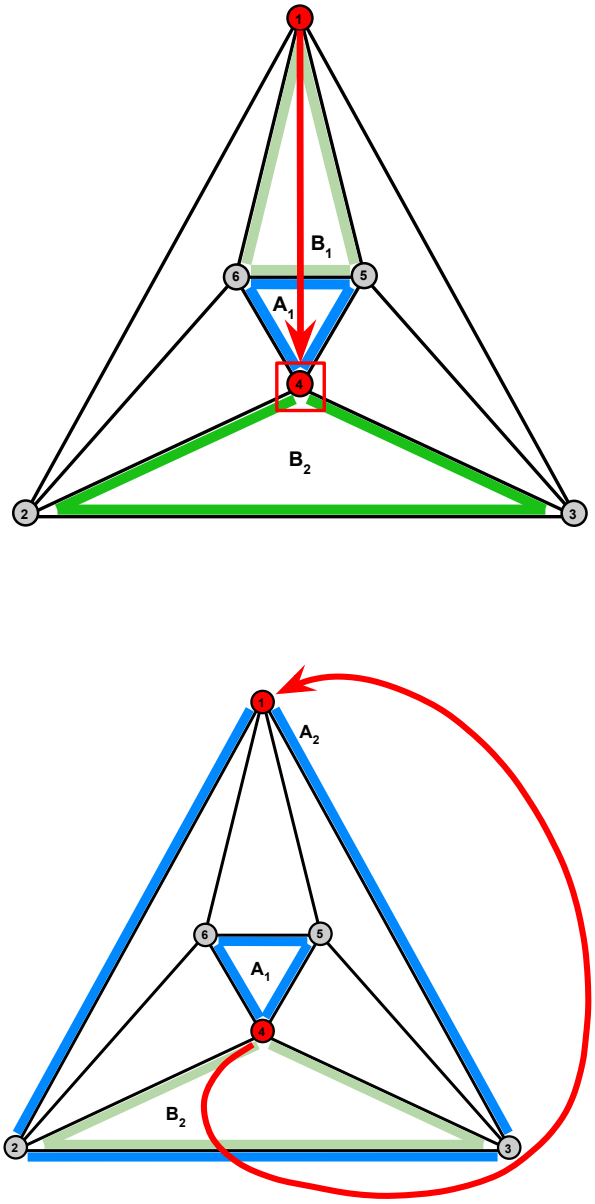


Figure 4.5 Top: Drop vertex 1 from room partition B. Pivot from room B_1 to room A_1. The new vertex 4 is duplicate in room A_1 and room B_2. Bottom: Drop duplicate vertex 4. Pivot from the old room B_2 to room A_2, picking up the missing vertex.

Végh and von Stengel proved that the endpoints of the Complementary Pivoting with Direction Algorithm have opposite orientation, as long as the room partitions are defined on an oriented oik; the parity result follows as in the previous ones. Unfortunately, as for the Lemke-Howson Algorithm, the Complementary Pivoting with Direction Algorithm may take exponential time in the general case. Despite this, Végh and von Stengel [25] give a near-linear-time algorithm that, given a perfect matching of an Euler graph, finds a perfect matching of opposite sign; this allows to find a solution to Opposite Sign Gale in polynomial time.

The wider issue of the complexity of the Lemke-Howson Algorithm has been solved by Goldberg, Papadimitriou and Savani [12] as **PSPACE**-complete. This invites, once more, a further investigation of exchange-like algorithms to build "hard to solve" games. Since restricting to games that can be reduced to oiks of dimension 2 seems to give games that are, after all, easily solvable, a possible direction for research could be the study of games built on oiks of dimension 3 or higher; studying products of these could also be interesting, although the use of products of polytopes to make the solvability via enumeration support harder in Savani and von Stengel [22] was circumvented together with the simpler case of Gale games by our result.

Another result worth mentioning is found in Merschen [16]: finding the number of equilibria of a Gale game is #**P**-complete. Gilboa and Zemel [11] have proven that deciding the uniqueness of Nash equilibria is a **co-NP**-complete problem; a proof from the point of view of completely labeled facets of a polytope was given by von Stengel [28] with the result of **NP**-completeness of the problem of finding a completely labeled facet in a generic labeled polytope. Given these results, it could be interesting to find a polynomial-time algorithm to decide the uniqueness of Nash equilibria in Gale games, or to find yet another watershed for the complexity of games in the more general framework described above.

Acknowledgements

appendix/acknowledgments

Bibliography

[1] A. V. Balthasar (2009). "Geometry and equilibria in bimatrix games." PhD Thesis, London School of Economics and Political Science.

[2] M. M. Casetti (2008). "PPAD Completeness of Equilibrium Computation." MSc Thesis, London School of Economics and Political Science.

[3] M. M. Casetti, J. Merschen, B. von Stengel (2010). "Finding Gale Strings." *Electronic Notes in Discrete Mathematics* 36, pp. 1065–1082.

[4] X. Chen, X. Deng (2006). "Settling the Complexity of 2-Player Nash Equilibrium." *Proc. 47th FOCS*, pp. 261–272.

[5] C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou (2009). "The Complexity of Computing a Nash Equilibrium." *SIAM Journal on Computing*, 39(1), pp. 195–259.

[6] J. Edmonds (1965). "Paths, Trees, and Flowers." *Canad. J. Math.* 17, pp. 449–467.

[7] J. Edmonds (2007). "Euler complexes." *www.imada.sdu.dk/asp/edmonds.pdf*

[8] J. Edmonds, L. Sanità (2010). "On finding another room-partitioning of the vertices." *Electronic Notes in Discrete Mathematics* 36, pp. 1257–1264.

[9] D. Gale (1963). "Neighborly and Cyclic Polytopes." *Convexity, Proc. Symposia in Pure Math.*, Vol. 7, ed. V. Klee, American Math. Soc., Providence, Rhode Island, pp. 225–232.

[10] D. Gale, H. W. Kuhn, A. W. Tucker (1950). "On Symmetric Games." *Contributions to the Theory of Games* I, eds. H. W. Kuhn and A. W. Tucker, *Annals of Mathematics Studies* 24, Princeton University Press, Princeton, pp. 81–87.

[11] I. Gilboa, E. Zemel (1989). "Nash and correlated equilibria: some complexity considerations." *Games and Economic Behavior* 1, pp. 80–93.

[12] P. W. Goldberg, C. H. Papadimitriou, R. Savani (2011). "The Complexity of the Homotopy Method, Equilibrium Selection, and Lemke-Howson solutions." *IEEE Symposium on Foundations of Computer Science (FOCS).*

[13] C. E. Lemke, J. T. Howson, Jr. (1964). "Equilibrium Points of Bimatrix Games." *J. Soc. Indust. Appl. Mathematics* 12, pp. 413–423.

[14] A. McLennan, R. Tourky (2010). "Imitation Games and Computation." *Games and Economic Behavior* 70, pp. 4–11.

[15] N. Megiddo, C. H. Papadimitriou (1991). "On Total Functions, Existence Theorems and Computational Complexity." *Theoretical Computer Science* 81, pp. 317–324.

[16] J. Merschen (2012). "Nash Equilibria, Gale Strings, and Perfect Matchings." PhD Thesis, London School of Economics and Political Science.

[17] W. D. Morris Jr. (1994). "Lemke Paths on Simple Polytopes." *Math. Oper. Res.* 19, pp. 780–789.

[18] J. F. Nash (1951). "Noncooperative games." *Annals of Mathematics*, 54, pp. 289–295.

[19] M. J. Osborne, A. Rubinstein (1994). *A Course in Game Theory.* The MIT Press, Cambridge, Massachusetts.

[20] C. H. Papadimitriou (1994). *Computational Complexity.* Addison-Wesley, Reading, MA.

[21] C. H. Papadimitriou (1994). "On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence." *J. Comput. System Sci.* 48, pp. 498–532.

[22] R. Savani, B. von Stengel (2006). "Hard-to-solve Bimatrix Games." *Econometrica* 74, pp. 397–429.

[23] R. Savani, B. von Stengel (2015). "Unit Vector Games." arXiv:1501.02243v1 [cs.GT]

[24] L. S. Shapley (1974). "A Note on the Lemke-Howson Algorithm." *Mathematical Programming Study 1: Pivoting and Extensions*, pp. 175–189

[25] L. Végh, B. von Stengel "Oriented Euler Complexes and Signed Perfect Matchings." arXiv:1210.4694v2 [cs.DM]

[26] J. von Neumann, (1928). "Zur Theorie der Gesellschaftspiele." *Mathematische Annalen*, 100, pp. 295–320.

[27] B. von Stengel (2007). "Equilibrium computation for two-player games in strategic and extensive form." Chapter 3, "Algorithmic Game Theory," eds. N. Nisan,

T. Roughgarden, E. Tardos, V. Vazirani. Cambridge Univ. Press, Cambridge, pp. 53–78.

[28] B. von Stengel (2012). "Completely Labeled Facet is NP-Complete." Manuscript, 6 pp.

[29] G. M. Ziegler (1995) *Lectures on Polytopes.* Springer, New York.