

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 [4]. The improvement given by Proposition 2.4 is an original result.

The copyright of this thesis rests with the author. Quotation from it is permitted, provided that full acknowledgement is made. This thesis may not be reproduced without my prior written consent.

I warrant that this authorisation does not, to the best of my belief, infringe the rights of any third party.

Abstract

This thesis presents a report on original research, published as joint work with Merschen and von Stengel in *Electronic Notes in Discrete Mathematics* [4]. Our result presents a polynomial time algorithm to solve two problems related to labeled Gale strings, a combinatorial structure introduced by Gale in [10] that can be used in the representation of a particular class of games.

These games were used by Savani and von Stengel [24] as an example of exponential running time for the classical Lemke-Howson algorithm to find a Nash equilibrium of a bimatrix game [15]. It was therefore conjectured that solving these games was a problem complete for the class **PPAD** (Polynomial Parity Argument, Directed version, see Papadimitriou [23]). In turn, a major motivation for the definition of PPAD was to study complementary pivoting methods, including the Lemke-Howson algorithm.

Our result, on the contrary, sets apart this class of games as a case where a Nash equilibrium can be found in polynomial time. Since Daskalakis, Goldberg and Papadimitriou [6] and Chen and Deng [5] proved the **PPAD**-completeness

of finding a Nash equilibrium in general normal-form games, we have a special class of games, unless $\mathbf{PPAD} = \mathbf{P}$.

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 [24] [25]; then, we use the polynomial-time solvability of the problem of finding a perfect matching in a graph, proven by Edmonds [7].

Merschen [18] and Végh and von Stengel [27] expanded our ideas to prove further results concerning the index of a Nash equilibrium, as defined in Shapley [26], in the framework of “oiks” as considered by Edmonds [8] and Edmonds and Sanità [9].

Contents

1	Introduction	9
1.1	Vectors and Polytopes	10
1.2	Normal Form Games and Nash Equilibria	12
1.3	Computational Problems and Complexity	14
2	Labels, Polytopes and Gale Strings	20
2.1	Bimatrix Games and Labels	21
2.2	Cyclic Polytopes and Gale Strings	32
2.3	Gale Games	38
3	Algorithmic and Complexity Results	42
3.1	Polynomial Parity Argument	43
3.2	The Lemke-Howson Algorithm	46
3.3	The Complexity of ANOTHER GALE	60
4	Further results	69
	Acknowledgments	75
	Bibliography	79

List of Figures

2.1	The labeled mixed strategy simplices of a game	23
2.2	The best response polyhedron of a game	24
2.3	The best response polytope of a game	25
2.4	The best response regions of a symmetric game	27
2.5	A degenerate symmetric game	28
2.6	The polytope P^l of a unit vector game	30
2.7	The cyclic polytope $C_3(6)$	33
2.8	A facet of the cyclic polytope $C_3(6)$	36
2.9	A facet of $C_3(6)$ as zeroes of the moment curve	36
2.10	The cyclic polytope $C_4(6)$	37
2.11	A facet of $C_4(6)$ via the moment curve	37
2.12	Not a facet of $C_4(6)$	38
2.13	A labeling of $C_4(6)$ and its completely labeled facets	40
3.1	A PPAD problem	45
3.2	A pivot on the vertices of the cube	47
3.3	A pivot on the facets of the octahedron	47
3.4	A Lemke path for a bimatrix game	51
3.5	A game with disjoint Lemke paths	52
3.6	A pivot on $G(4, 6)$ and on $C_4(6)$	56
3.7	Pivoting with sign	58
3.8	The Morris graph for $G(6, 4)$	64

3.9	A graph without a perfect matching	64
3.10	Finding a second perfect matching	67
3.11	The second matching of the Morris graph	68
4.1	The endpoints of the Dual Lemke Path Algorithm	71
4.2	The Exchange Algorithm	72
4.3	The endpoints of the Exchange Algorithm	73

List of Tables

1.1	The prisoners' dilemma	13
1.2	A coordination game	14
1.3	n -NASH	15
2.1	ANOTHER COMPLETELY LABELED VERTEX	32
2.2	ANOTHER COMPLETELY LABELED FACET	32
2.3	GALE NASH	39
2.4	ANOTHER GALE	41
3.1	END OF THE LINE	44
3.2	Sign switching on the Lemke Path for Gale Algorithm	57
3.3	A Lemke path for the Morris labeling	60
3.4	PERFECT MATCHING	61
3.5	GALE	61
4.1	OPPOSITE SIGN GALE	69

List of Algorithms

1	Lemke Path	49
2	Dual Lemke Path	53
3	Lemke Path for Gale	55
4	Find a Completely Labeled Gale String	63
5	Find Another Gale String	66

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 is 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 [10]), using a construction (Savani and von Stengel [24]) on labeled polytopes derived from the game itself (Lemke and Howson [15], Shapley [26]). This connection was used to construct games for which the classic Lemke-Howson Algorithm (Lemke and Howson [15]) takes exponential running time to find a Nash equilibrium (Savani and von Stengel [24]). 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** (Polynomial Parity Argument, Directed version; Papadimitriou [23]). Its **PPAD**-completeness has been proven (Daskalakis, Goldberg and Papadimitriou [6], Chen and Deng [5]), but the result required the use of approximate 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 $\textbf{PPAD} = \textbf{P}$, Gale games are indeed a case apart, but because of their tractability, not because 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 [31] for details), basic game theory (see Osborne and Rubinstein [21]) and computational complexity (see Papadimitriou [22]). 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 complexity 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 vector 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. A *convex combination* of the points z_i is such an affine combination with $\lambda_i \geq 0$ for all $i \in [m]$. The *convex hull* of a set Z of points is the set of all its convex combinations. We denote it as $\text{conv}(Z)$, so

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

A set of points Z is *convex* if $Z = \text{conv}(Z)$. It has *dimension* d if it contains exactly $d + 1$ affinely independent points. The convex hull 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 a convex set. A *polyhedron* is the intersection of finitely many closed halfspaces $\{x \in \mathbb{R}^d \mid a^\top x \leq a_0\}$. It can be shown that a bounded polyhedron is a polytope. A *vertex* of a d -dimensional polytope $P = \text{conv}(Z)$ is a point $z \in Z$ that is not the convex combination of other points in P . An *edge* of P is a 1-dimensional line segment that has two vertices as endpoints. A *facet* of P is a set of dimension $d - 1$ that is the convex hull of a set of vertices that lie on a hyperplane of the form $\{x \in \mathbb{R}^d \mid a^\top x = a_0\}$ so that $a^\top u < a_0$ 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 d facets. Notice that the points lying on exactly d facets are exactly the vertices. Consider a polytope P which has $\mathbf{0}$ in its interior and is given by n inequalities with normal vectors c_i for $i \in [n]$ according to

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

Then the *polar* of the polytope P is denoted by P^Δ , where

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

If P is a polytope and contains the origin $\mathbf{0}$, the polar of its polar is P itself, $P^{\Delta\Delta} = P$. Furthermore, if P is simplicial then P^Δ is simple, and vice versa. For details see Ziegler [31].

1.2 Normal Form Games and Nash Equilibria

The idea of *game* as model of strategic interaction was first introduced by von Neumann [28]. A *finite normal form game* is $\Gamma = (P, S, u)$ where both P and S are finite. Here P is the set of *players*, and $S = \times_{p \in P} S_p$ is the set of *pure strategy profiles*, where S_p is the set of *pure strategies* of player p . The aim of each player $p \in P$ is to maximize their *payoff function* $u^p : S \rightarrow \mathbb{R}$. The vector of payoffs is $u = \times_{p \in P} u^p$. 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 $u^p : \Delta \rightarrow \mathbb{R}$ 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*. Notice that applying a positive affine transformation to all the payoffs does not change the Nash equilibria of the game. The following “best-response condition” is useful and easy to show (e.g., Nash [20]).

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 ([20]). Notice that there can be more than one equilibrium.

Theorem 1. (Nash [20]) *Every finite game in normal form has at least one Nash equilibrium.*

Bimatrix games are games with only two players. They can be described by two $m \times n$ 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 i th pure strategy and the latter plays his j th pure strategy, and $m = |S_1|$ and $n = |S_2|$. 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.

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

Table 1.1 The prisoners' dilemma.

Table 1.2 shows a *coordination game*. Both players drive on either a mountain or a valley road. They lose if they drive on the same road and win if they avoid each other, regardless of which road they take. The pure strategy Nash

equilibria are (mountain, valley) and (valley, mountain). There is also a symmetric equilibrium in mixed strategies (represented as vectors of probabilities) at $((1/2, 1/2), (1/2, 1/2))$.

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

Table 1.2 A coordination game.

1.3 Computational Problems and Complexity

We base this section on the definitions in Papadimitriou [22], to which we refer for further details.

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 algorithm manipulating the input on a *tape*, and returns an *output* when it comes to a halting state (if it ever does).

The tape and its manipulation can be seen as follows: initially the tape contains the input, given by a *first symbol* \triangleright followed by a finite string of symbols $x \in (\Sigma \setminus \{\sqcup\})^*$, where Σ is the *alphabet* of symbols of \mathcal{M} and \sqcup is a special symbol representing the *blank* string. The program is then carried on: a *cursor* starts at the first symbol and follows the *algorithm* given by a *transition function* δ . The transition function depends on the current *state* $p \in K$ and the current symbol $\sigma \in \Sigma$: the former is the “instruction” to follow, the latter the symbol in the position of the cursor. The transition function returns the next state $q \in K$, the symbol $\tau \in \Sigma$ to be written at the position of the cursor, and a direction in which the cursor will move on the tape. The

latter can be “left”, “right”, or “stay”, represented respectively by \leftarrow , \rightarrow , $-$. The first symbol is never overwritten, and from there the cursor can only go left.

In some cases, after some finite time, the machine reaches the state YES (the machine *accepts* the input), NO (the machine *rejects* the input) or the *halting state* h . The output is then given by $\mathcal{M}(x) = \text{YES}$ or $\mathcal{M}(x) = \text{NO}$ if the machine accepts or rejects the input. If the halting state h is reached, the output is $\mathcal{M}(x) = y$, where y is the finite (possibly empty) string of symbols that is left on the tape after the symbol \triangleright and before any string of \sqcup 's at the end. It is also possible that the machine doesn't halt on input x ; we denote this case as $\mathcal{M}(x) = \nearrow$.

A problem P that requires an output that is 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 problem where the output can be a string $\mathcal{M}(x) = y$ on the output tape as above is a *function problem*; note that no guarantee on the halting state is required. If the output is a string $\mathcal{M}(x) = y$ satisfying a relation $R(x, y)$ or the input is rejected if it is not possible to find any such string, the problem is a *search problem*. A search problem where the input is never rejected 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.

Formally, $\mathcal{M} = (K, \Sigma, \delta, s)$, where K and Σ are finite, $\Sigma \cap K = \emptyset$ and Σ always contains the symbols \sqcup and \triangleright . The *transition function* is δ , defined as follows, where $\{\leftarrow, \rightarrow, -\} \not\subseteq (K \cup \Sigma)$:

$$\delta : K \times \Sigma \longrightarrow (K \cup \{h, \text{ YES}, \text{ NO}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}.$$

A slightly different, although equivalent, model allows for *multiple strings*: $k \in \mathbb{N}$ cursors move on k strings $\sigma_i \in \Sigma^*$; the states are still represented as $p \in K$. The input is given in the tape of the first string, the output in the tape of the k -th string (assuming the machine halts).

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 $\mathcal{M}(x) = \text{YES}$ if $x \in L$ or $\mathcal{M}(x) = \text{NO}$ if $x \notin L$. If for every $x \in (\Sigma \setminus \{\sqcup\})^*$ either $\mathcal{M}(x) = \text{YES}$ if and only if $x \in L$, or $\mathcal{M}(x) = \text{NO}$ or \mathcal{M} does not halt if and only if $x \notin L$, the Turing machine \mathcal{M} *accepts* L . Finally, \mathcal{M} *computes* a function $f : (\Sigma \setminus \{\sqcup\})^* \rightarrow \Sigma^*$ if $\mathcal{M}(x) = f(x)$ for every $x \in (\Sigma \setminus \{\sqcup\})^*$.

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 $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$, a Turing machine \mathcal{M} , and a polynomial p 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.

For any class C of decision problems, the class of all complements of the problems in C is the *complement class* $\text{co} - C$. 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 at least as hard to solve as every problem in C . A *complete* problem for the class C is a $C - \text{hard}$ 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** (*polynomial-time*) 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*) 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 .

An equivalent definition gives **NP** as the class of all languages L for which the binary relation $R(x,y)$ such that $L = \{x \mid R(x,y) \text{ holds for some } y\}$ satisfies the following conditions:

1. (*polynomially balanced*) if $(x,y) \in R$ then $|y| \leq |x|^k$ for some $k \in \mathbb{N}$;
2. (*polynomially decidable*) if there is a Turing machine that decides L in polynomial time.

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” y is, indeed, a correct answer to the question posed by the problem. Notice that there may be many different certificates for each instance of a problem. Consider for instance the **NP** problem HAMILTON PATH, defined as “given an input graph x , is it possible to find a Hamiltonian path y of x ?”. There are graphs with more than one possible Hamiltonian path, and each one of these can be used as certificate.

It is quite simple to see that $\mathbf{P} \subseteq \mathbf{NP}$, but it is an important open problem whether the inclusion is strict. If this were not the case, it could be argued

that there isn't any substantial difference between finding a solution and verifying its validity. This seems to contradict most intellectual experience, where the value of an “original” discovery is perceived as fundamental; the “conventional” view is therefore that $\mathbf{P} \neq \mathbf{NP}$. Another open problem is whether $\mathbf{NP} = \mathbf{co-NP}$; again, this is widely believed to be false. Notice that if $\mathbf{NP} \neq \mathbf{co-NP}$ then also $\mathbf{P} \subsetneq \mathbf{NP}$, but not vice versa.

The classes **FP** and **FNP** are analogous to **P** and **NP**, respectively, but for function problems instead of decision problems. Formally, **FNP** (*function non-deterministic polynomial-time*) is defined in Megiddo and Papadimitriou [17] as the class of problems of the form “given an $x \in \Sigma^*$, find $y \in \Sigma^*$ such that $(x, y) \in R$, where R is a polynomially balanced binary relation, or reject the input.” **FP** (*function polynomial-time*) is the class of all **FNP** problems that can be solved in polynomial time. As in the case of decision problems, it is not known whether $\mathbf{FP} = \mathbf{FNP}$; the question is actually equivalent to whether $\mathbf{P} = \mathbf{NP}$. Megiddo and Papadimitriou [17] also define the class **TFNP** (*total function non-deterministic polynomial-time*) as the class of all **FNP** problems where for every $x \in \Sigma^*$ a solution $y \in \Sigma^*$ is guaranteed to exist. The class **TFNP** is equivalent to $\mathbf{NPco-NP}$ for function problems, and the existence of a **TFNP**-complete problem would imply that $\mathbf{NP} = \mathbf{co-NP}$. The lack of **TFNP**-complete problems has led to define a number of new classes; we will see more of this in section 3.1.

The class $\#\mathbf{P}$ is the class of all function problems that output the number of possible certificates for a decision problem in **NP**. Formally: the *counting problem* associated with a binary relation $Q(x, y)$ is “given x , how many y are there such that $(x, y) \in Q$?” Then $\#\mathbf{P}$ is the class of all counting problems associated with binary relations that are both polynomially balanced and polynomially decidable. It is interesting to notice that there are $\#\mathbf{P}$ -complete problems for which the corresponding search problem can be solved in polynomial time. An example close to the topic of this thesis is given in Brightwell

[2]: although finding an Eulerian circuit in an undirected graph takes polynomial time, counting the number of possible circuits in the same graph is complete in $\#\mathbf{P}$.

Finally, the class **PSPACE** is the set of decision problems that can be solved by a Turing machine using an amount of tape space that is polynomial in the size of the input; although it can be proven that $\mathbf{NP} \subseteq \mathbf{PSPACE}$, the possibility of an identity is yet another open problem.

Chapter 2

Labels, Polytopes and Gale Strings

In the remainder of this thesis we focus on bimatrix games. We identify the game with its payoff matrices (A, B) , and we 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, by adding a constant to all entries in A and B if necessary.

We start by showing a construction to represent a bimatrix game as 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 [15]. We follow the very clear approach given by Shapley [26]. From labeled regions we then move on to an equivalent formulation in terms of “best response” polytopes; a result by McLennan and Tourky [16] 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 [25], allows us to simplify this construction. The problem 2-NASH can 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 restrict our scope to Gale games. These are unit vector games for which the polytopes in the previous section can be represented by combinatorial structures called Gale strings. We call the problem of finding a Nash equilibrium for a Gale game GALE NASH, and we will reduce it to a problem on Gale strings.

2.1 Bimatrix Games and Labels

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

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

A *labeling* of the sets X and Y assigns to each $x \in X$ and $y \in Y$ a set of *labels*, which is a subset of $[m+n]$, 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: A pair (x, y) is called *completely labeled* if each possible label in $[m+n]$ is a label of x or of y .

Theorem 2. (Shapley [26].) *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 $j \in [n]$, appear either as labels of x or of y . Conversely, if (x, y) is not completely labeled, then some label does not appear as a label of x or y . This label represents a pure strategy that is played with positive probability but is not a best response, which contradicts the equilibrium property. \square

A 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 into closed polyhedral sets, called *best response regions*, that correspond to the other player's pure best responses. We give an example of this construction.

Example 2.1. (Savani and von Stengel [25].) Consider the 3×3 game (A, B) with

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}. \quad (2.2)$$

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 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

$x = (\frac{1}{3}, \frac{2}{3}, 0)$ with labels 3, 4, 5, and $y = (\frac{1}{2}, \frac{1}{2}, 0)$ with labels 1, 2, 6. This is the only Nash equilibrium of the game.

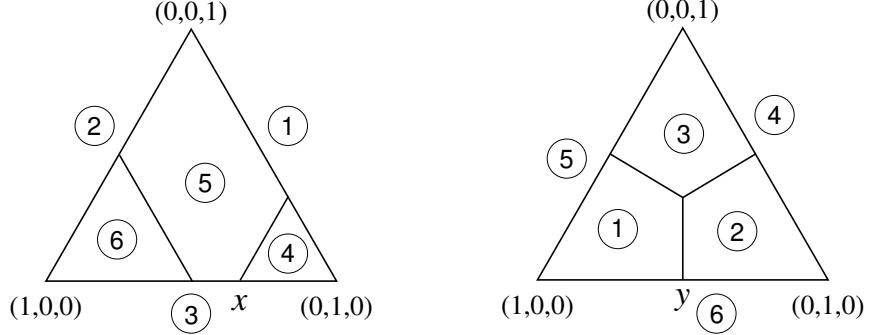


Figure 2.1 The labeled best response regions of the mixed strategy simplices of game (2.2).

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

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

In \bar{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 ; this payoff is the best-response payoff by the definition of P . The projection of the facet defined by $(B^\top x)_j = v$ to X then has label j . Analogously, the facet of \bar{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. (Savani and von Stengel [25].) 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 \bar{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.

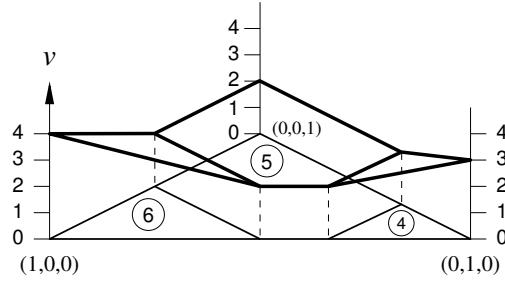


Figure 2.2 The best response polyhedron of game (2.2).

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 \bar{P} and \bar{Q} with the *best-response polytopes*

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

In (2.4), both P and Q are defined by $m+n$ inequalities that correspond, in the listed order, to the pure strategies of the two players with labels in $[m+n]$. The polytope P is the intersection of the $m+n$ half-spaces that correspond to either player 1 not playing her i -th pure strategy or to a best response j of player 2, where $i \in [m]$ and $j \in [n]$. The analogous statement 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 $(B^\top x)_j = 1$ for $k = m+j$ with $j \in [n]$, and a point in Q has label k if and only if either $(Ay)_k = 1$ for $k \in [m]$ or $y_j = 0$ for $k = m+j$ with $j \in [n]$.

Hence, a point $(x, y) \in P \times Q$ is completely labeled if and only if it satisfies the *complementarity condition* that states that for all $i \in [m]$ and all $j \in [n]$,

$$\begin{aligned} x_i = 0 &\quad \text{or} \quad (Ay)_i = 1 , \\ y_j = 0 &\quad \text{or} \quad (B^\top x)_j = 1 . \end{aligned} \tag{2.5}$$

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 refer to the latter case as *artificial equilibrium*.

Example 2.3. (Savani and von Stengel [25].) 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 $(x, y) = (\mathbf{0}, \mathbf{0})$ is completely labeled, since it is labeled by the labels 1, 2, 3 in P and 4, 5, 6 in Q .

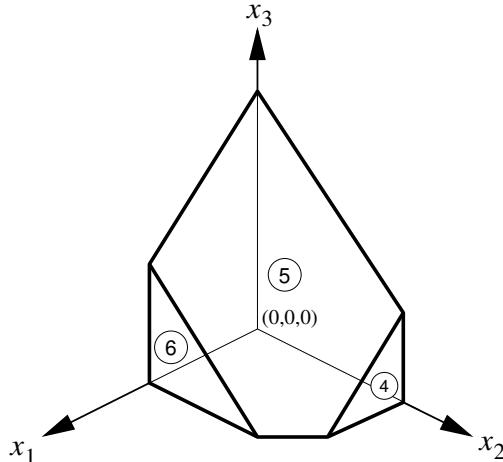


Figure 2.3 The best response polytope P of game (2.2).

We now consider 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”. This result is due to Gale, Kuhn and Tucker [11] 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 $z = (x\alpha, y\beta)$ for suitable positive scalars α and β , is a Nash equilibrium of the symmetric game (C, C^\top) , where*

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

McLennan and Tourky [16] 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 algorithm 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 [16].) *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. (Savani and von Stengel [25].) As an example, consider the symmetric game (C, C^\top) with

$$C = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 2 & 2 \\ 4 & 0 & 0 \end{pmatrix}, \quad C^\top = \begin{pmatrix} 0 & 2 & 4 \\ 3 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}. \quad (2.7)$$

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 $y = (\frac{1}{2}, \frac{1}{2}, 0)$.

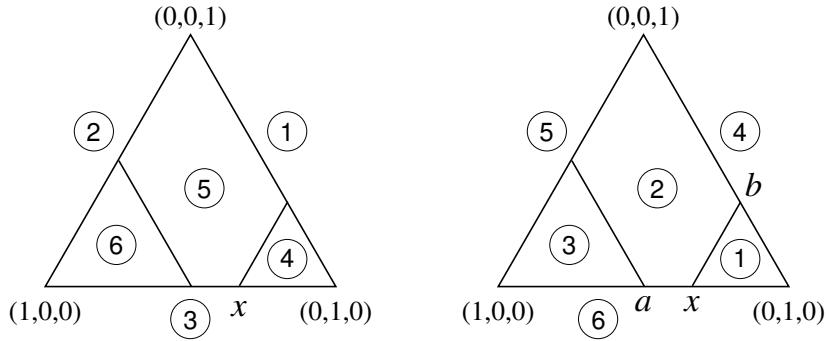


Figure 2.4 The best response regions of the symmetric game (2.7).

The characterization of Nash equilibria as completely labeled pairs (x, y) holds for arbitrary bimatrix games. From now on, we impose a further condition: all points in P have at most m labels, and all points in Q have at most n labels. These games are called *nondegenerate*. This condition is required for the Lemke-Howson algorithm (see below). The algorithm can also be applied to degenerate games by lexicographic perturbation (see von Stengel [29]). 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. (Savani and von Stengel [25].) An example of degenerate game is given by (C, C^\top) with

$$C = \begin{pmatrix} 0 & 4 & 0 \\ 2 & 2 & 2 \\ 4 & 0 & 0 \end{pmatrix}, \quad C^\top = \begin{pmatrix} 0 & 2 & 4 \\ 4 & 2 & 0 \\ 0 & 2 & 0 \end{pmatrix}. \quad (2.8)$$

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 .

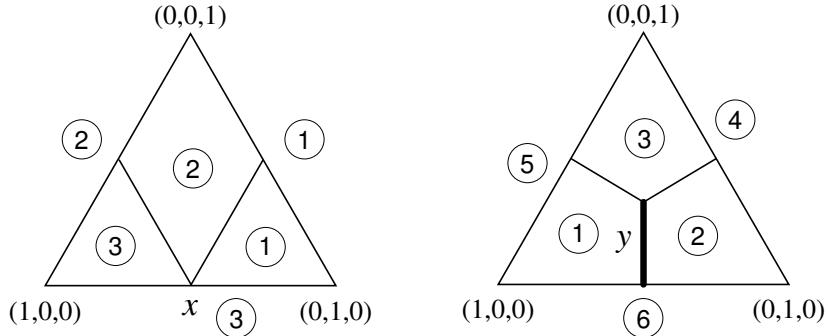


Figure 2.5 Left: The best-response regions of the degenerate symmetric game (2.8). Right: The best-response regions of the imitation game (I, C^\top) .

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 [25] 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 a single polytope P^l with a function l that labels the last n facets of P with elements in $[m]$ (instead of elements $m + j$ for $j \in [n]$) that encodes all information about the game.

Theorem 4. (Savani and von Stengel [25].) *Let $l : [n] \rightarrow [m]$ be a function, 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*

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

Let l_f be the labeling of the facets of P^l with labels in $[m]$ 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 $P^l \setminus \{\mathbf{0}\}$ (that is, has all labels in $[m]$) 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 with labels 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 $(B^\top x)_j = j$ and $l(j) = i$; that is, $j \in N_i$. For all $i \in [m]$ such that $x_i > 0$, define y as follows: $y_j = 1$, and $y_h = 0$ for all $h \in N_i \setminus \{j\}$. Then $(x, y) \in P \times Q$ is completely labeled. \square

Example 2.6. (Savani and von Stengel [25].) 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).

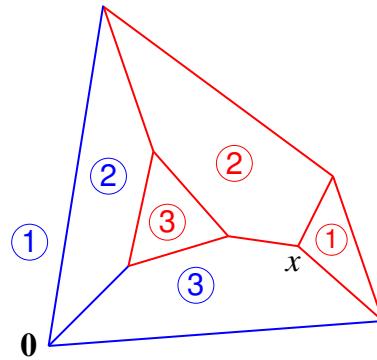


Figure 2.6 The polytope P^l of the unit vector game (2.2).

We now move on the dual version of Theorem 4, as given by Balthasar [1]. We can translate 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 , which holds if $\mathbf{1}$ is in the interior of P^l , that is, all columns b_j of the matrix B fulfill $\mathbf{1}^\top b_j < 1$, for $j \in [n]$. Then

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

The polar of P is then

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

where $c_j = b_j / (1 - \mathbf{1}^\top b_j)$. Since P and P^Δ have $\mathbf{0}$ in their interior, $P^{\Delta\Delta} = P$. Furthermore, P^Δ is simplicial and its facets correspond to the vertices of P

and vice versa. We label the vertices of P^Δ as the corresponding facets in P^l , so the completely labeled facets of P^Δ 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 to the following.

Theorem 5. (Balthasar [1].) *Let P^Δ be a labeled m -dimensional simplicial polytope with $\mathbf{0}$ in its interior and vertices $e_1, \dots, e_m, c_1, \dots, c_n$ such that F_0 in (2.12) is a facet of P^Δ . Let (U, B) be a unit vector game, with $U = (e_{l(1)} \cdots e_{l(n)})$ for a labeling $l : [n] \rightarrow [m]$ 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 P^Δ given by*

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

Then a facet $F \neq F_0$ of P^Δ 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 P^Δ 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 takes 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 : An m -dimensional simple polytope P with $m+n$ facets; a labeling $l_f : [m+n] \rightarrow [n]$; 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 P^Δ with $m+n$ vertices; a labeling $l_v : [m+n] \rightarrow [n]$; a facet F_0 of P^Δ that is completely labeled by l_v .
output: A facet $F \neq F_0$ of P^Δ 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 define cyclic polytopes, then Gale string, then

we will give the theorem by Gale [10] that shows the equivalence of the two representations.

The *moment curve* in dimension d is defined as

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

The *cyclic polytope* $C_d(n)$ in dimension d with n vertices, where $n > d$, is given as the convex hull of any n points on the moment curve, that is, by n arbitrary reals t_1, \dots, t_n , where $t_1 < \dots < t_n$, according to

$$C_d(n) = \text{conv}\{\mu_d(t_i) \mid 1 \leq i \leq n\}. \quad (2.15)$$

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

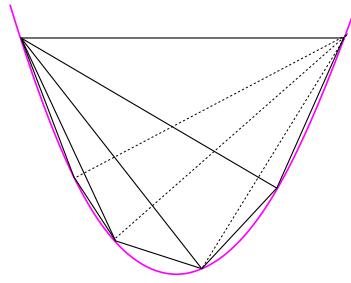


Figure 2.7 The cyclic polytope $C_3(6)$.

Given $k \in \mathbb{N}$ and a set S , we can represent the function $f : [k] \rightarrow S$ 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*. A run is called even if its length is even, and odd if its length is odd. 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 that satisfies the following conditions:

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

Let $G(d, n)$ be the set of these Gale strings.

In general, the Gale Evenness Condition 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 “gluing together” the endpoints of the strings; then all runs on the loops are even. Formally, we can see the bit positions in a Gale string $s \in G(d, n)$ with d even as equivalence classes modulo n .

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

$$G(4, 6) = \{\mathbf{111100}, \mathbf{111001}, \mathbf{110011}, \mathbf{100111}, \mathbf{001111}, \\ \mathbf{011110}, \mathbf{110110}, \mathbf{101101}, \mathbf{011011}\}$$

The strings $\mathbf{111100}$, $\mathbf{111001}$, $\mathbf{110011}$, $\mathbf{100111}$, $\mathbf{001111}$ and $\mathbf{011110}$ are equivalent under a cyclic shift (if considering the strings as “loops”, the $\mathbf{1}$ ’s are all consecutive), as are the strings $\mathbf{110110}$, $\mathbf{101101}$ and $\mathbf{011011}$ (two runs of two $\mathbf{1}$ ’s separated by a single 0). As an example for odd d , we have

$$G(3, 5) = \{\mathbf{11100}, \mathbf{10110}, \mathbf{10011}, \mathbf{11001}, \mathbf{01101}, \mathbf{00111}\}.$$

Notice that because d is odd, a cyclic shift is not allowed: $\mathbf{01011}$ is a shift of $\mathbf{10110}$ but it is not a Gale string.

The relation between cyclic polytopes and Gale strings was given by Gale [10].

Theorem 6. (Gale [10].) *For any $d, n \in \mathbb{N}$, where $n > d$, a set F is a facet of $C_d(n)$ if and only if*

$$F = \text{conv}\{\mu(t_j) \mid s(j) = 1 \quad \text{for } s \in G(d, n)\}. \quad (2.16)$$

Proof. First, a hyperplane in \mathbb{R}^d of the form $\{x \in \mathbb{R}^d \mid a^\top x = a_0\}$ for some nonzero vector $a = (a_1, \dots, a_d)^\top$ can contain at most d points on the moment curve, because otherwise the polynomial equation with a polynomial of degree d given by $-a_0 + a_1 t + a_2 t^2 + \dots + a_d t^d = 0$ would have more than d

roots t . For the same reason, it is easy to see that any d points on the moment curve are affinely independent and define a unique hyperplane through them, which the moment curve *crosses* at these points (if it only touched the hyperplane, then a slight perturbation of the hyperplane would contain $d+1$ points on the moment curve). Let $\bar{t}_1 < \dots < \bar{t}_d$ be a choice of d of the t_j 's in the definition (2.15) of $C_d(n)$, and consider the hyperplane H that contains the points $\mu_d(\bar{t}_1), \dots, \mu_d(\bar{t}_d)$. The moment curve crosses H at these points. That is, if $t, t' \notin \{\bar{t}_i\}$ and $t < \bar{t}_i < t'$ for exactly one \bar{t}_i then $\mu_d(t)$ and $\mu_d(t')$ are on opposite sides of H . A facet F of the cyclic polytope $C_d(n)$ given by $F = H \cap C_d(n)$ then corresponds to a choice of \bar{t}_i 's such that for all the other t_k 's in the definition of $C_d(n)$, that is, such that $t_k \notin \{\bar{t}_i \mid i \in [d]\}$, the corresponding points $\mu_d(t_k)$ are on the *same* side of H . This can happen only if for every pair of these t_k 's the moment curve has an even number of crossings of H between them. Hence, there is an even number of \bar{t}_i 's between any two t_k 's. Let s be the bitstring in which the 1's correspond to the \bar{t}_i 's and the 0's correspond to the other t_k 's. Then the condition that the set F in (2.16) is a facet is equivalent to the Gale Evenness Condition. \square

Because the moment curve has at most d points on any hyperplane, each facet of $C_d(n)$ is a d -simplex, so $C_d(n)$ is simplicial and the choice of the t_j 's in (2.15) is irrelevant for the characterization of the facets of $C_d(n)$ as Gale strings. However, note that the order of the points on the curve, that is, of $t_1 < \dots < t_n$, defines the positions in the bitstring.

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 $s = \mathbf{100110}$. 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 $s(i) = 1$.

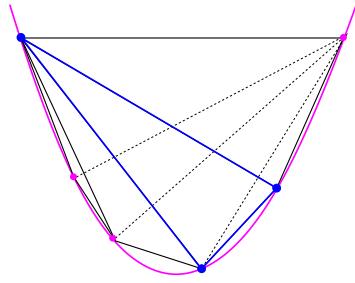


Figure 2.8 The facet of the cyclic polytope $C_3(6)$ corresponding to the Gale string $s = \mathbf{100110} \in G(3, 6)$.

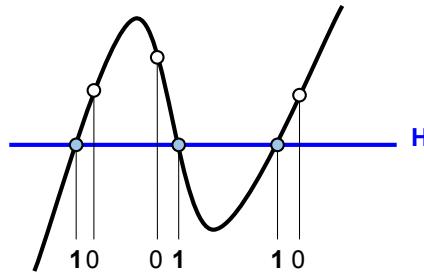


Figure 2.9 The facet of the cyclic polytope $C_3(6)$ given by the intersection of the moment curve and the hyperplane H (seen “from the side”) through the points $\mu_3(t_1), \mu_3(t_4), \mu_3(t_5)$ corresponds to the Gale string $s = \mathbf{100110} \in G(3, 6)$.

Example 2.10. Figure 2.10 shows the cyclic polytope $C_4(6)$, with the exterior facet corresponding to the Gale string $s = \mathbf{111100}$. Figure 2.11 shows the string $s = \mathbf{111100}$ as intersection of the moment curve and the hyperplane H .

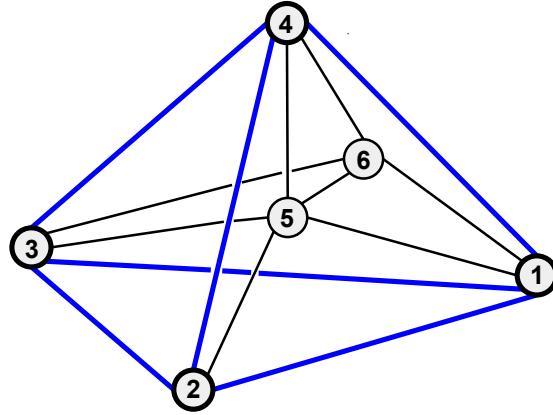


Figure 2.10 The cyclic polytope $C_4(6)$. The thin lines represent the edges inside the exterior facet, in bold lines; the latter corresponds to the Gale string $s = \mathbf{111100} \in G(4, 6)$.

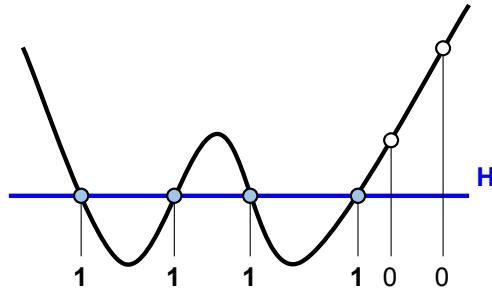


Figure 2.11 The facet of the cyclic polytope $C_4(6)$ given by the intersection of the moment curve and the hyperplane H through the points $\mu_4(t_1), \mu_4(t_2), \mu_4(t_3), \mu_4(t_4)$, corresponding to the Gale string $s = \mathbf{111100} \in G(4, 6)$.

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 $\mu_4(t_4)$ and $\mu_4(t_6)$ are on opposite sides of the hyperplane H defined by the other four points. The corresponding bitstring is $s = \mathbf{111010}$, which is not a Gale string. The violation of the Gale Evenness Condition corresponds to the change of side with respect to the hyperplane between the t_j 's.

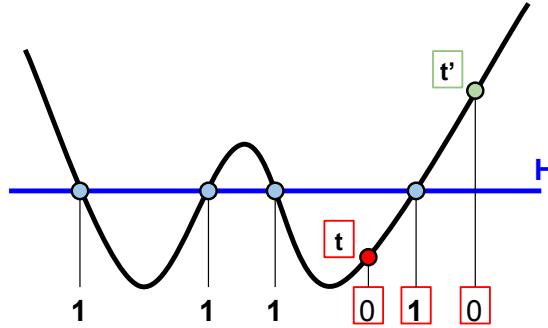


Figure 2.12 There is a change of side between two $\mu_4(t_j)$'s (for the 0 bits). The bitstring $s = \mathbf{111010}$ does not satisfy the Gale Evenness Condition, and the set of $\mu_4(\bar{t}_i)$'s (for the 1 bits) does not define a facet of $C_4(6)$.

2.3 Gale Games

We now apply Theorem 6 to the study of bimatrix games. By Proposition 2.2, solving 2-NASH can be reduced to ANOTHER COMPLETELY LABELED FACET. If the polytope P^Δ in Theorem 5 is cyclic, we can exploit Theorem 6 to translate this to a problem on Gale strings. First of all, we have to define a labeling for Gale strings such that a completely labeled Gale string corresponds to a completely labeled facet of P^Δ .

We say that $s \in G(d, n)$ is a *completely labeled Gale string* for some labeling function $l_s : [n] \rightarrow [d]$ if $\{l_s(i) \mid s(i) = \mathbf{1}\text{ for }i \in [n]\} = [d]$. Since $s \in G(d, n)$ has exactly 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 $l_s(i) = j$. It may not always be 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 $s_1 = \mathbf{1}1010\mathbf{1}$, $s_2 = 0\mathbf{1}110\mathbf{1}$, $s_3 = 010\mathbf{1}11$, but none of these satisfies the Gale Evenness Condition.

A *Gale game* is a unit vector game (U, B) where $U = [e_{l(1)} \cdots e_{l(d)}]$ for some labeling $l : [n] \rightarrow [d]$ and for which the dual of the best response polytope is a cyclic polytope $P^\Delta = \text{conv}\{e_1, \dots, e_d, c_1, \dots, c_n\}$. We define the problem GALE NASH as in Table 2.3

GALE NASH

input : A Gale game.

output: A Nash equilibrium of the game.

Table 2.3 The problem GALE NASH.

Theorem 5 gives the labeling (2.13) for the $d + n$ vertices of P^Δ as

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

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

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

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

From this point forward, we will assume that the labeling $l_s : [d + n] \rightarrow [d]$ 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** (it is possible that both are equal to 0), so $s(i)s(i+1)$ will never be part of 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 $G(4, 6)$: $s_A = \mathbf{111100}$, $s_B = \mathbf{110110}$, $s_C = \mathbf{100111}$ and $s_D = \mathbf{101101}$. These correspond to the completely labeled facets for the labeling shown in Figure 2.13 on the left.

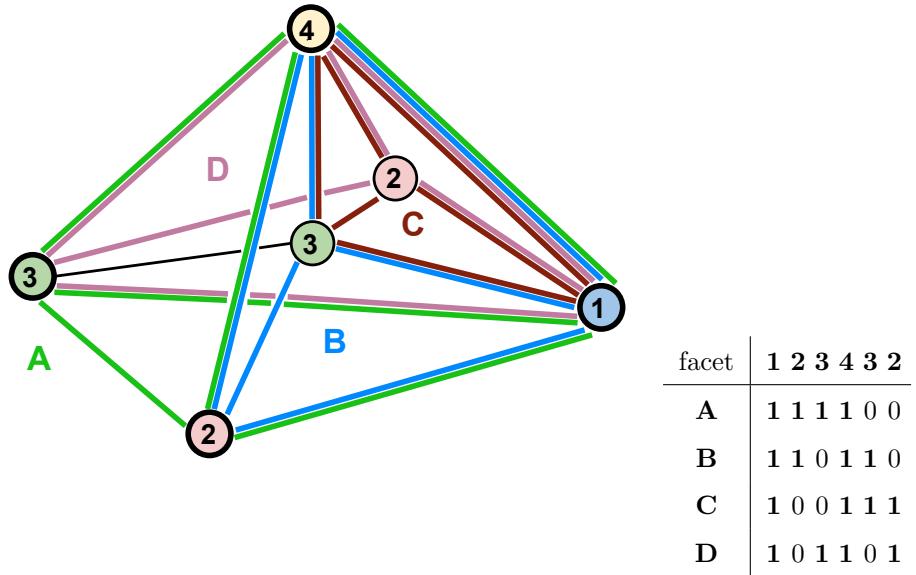


Figure 2.13 The cyclic polytope $C_4(6)$. The labeling of the vertices corresponds to the labeling of $G(4, 6)$ given by $l_s = 123432$.

We can now define the problem ANOTHER GALE as in Table 2.4, where for brevity we write n instead of $d+n$.

It takes polynomial time to translate the facets of the cyclic polytope $C_d(d+n)$ into the corresponding Gale strings in $G(d, d+n)$, following the proof of Theorem 6. 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

ANOTHER GALE

input : A labeling $l : [n] \rightarrow [d]$, where $d < n$. A Gale string $s \in G(d, n)$, completely labeled by l .

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

Table 2.4 The problem ANOTHER GALE.

$j \in [n]$, 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. *The problem GALE NASH of Table 2.3 is polynomial-time reducible to the problem ANOTHER GALE of Table 2.4.*

Proposition 2.3 can be improved: it is enough to consider the case where d is even.

Proposition 2.4. *The problem ANOTHER GALE of Table 2.4 is reducible to the case where d is even.*

Proof. Consider an instance of the problem ANOTHER GALE with d odd.

Let $s'_0(i) = s_0(i)$ for $i \in [n]$ and $s'_0(n+1) = 1$, and let $l' : [n+1] \rightarrow [d+1]$ be the labeling such that $l'(i) = l(i)$ for $i \in [n]$ and $l'(n+1) = d+1$.

Let s' be a bitstring of length $n+1$ such that $s'(n+1) = 1$, and let s be the bitstring of length n such that $s(i) = s'(i)$ for $i \in [n]$. First of all, notice that if s' is a Gale string then s is also a Gale string: the Gale Evenness Condition holds in all the interior runs, which is the only requirement for a Gale string in $G(d, n)$ when d is odd. Furthermore, if s' is completely labeled for l' , then $s'(n+1) = 1$, since the only occurrence of label $d+1$ is at index $n+1$.

Therefore, a solution for the original instance of ANOTHER GALE can be found by solving the problem ANOTHER GALE with input s'_0 and l' and output s' , then considering the corresponding completely labeled Gale string $s \in G(d, n)$ defined as above. \square

Chapter 3

Algorithmic and Complexity Results

In the previous chapter we have defined some problems of the form “find another completely labeled” vertex, facet or Gale string. In this chapter we finally study the complexity of these problems. In the first section we give the definition of the classes **PPAD** and **PPA**, first introduced by Papadimitriou [23]. In the second section we present different versions of the standard algorithm introduced by Lemke and Howson [15]. We apply it to ANOTHER COMPLETELY LABELED VERTEX and to find a solution for 2-NASH, its original motivation. We then study the Lemke Path for Gale Algorithm, which returns a solution to ANOTHER GALE. We use these algorithms to prove the **PPA** complexity of ANOTHER COMPLETELY LABELED VERTEX and ANOTHER COMPLETELY LABELED FACET. In the case of ANOTHER GALE we extend the proof to **PPAD** complexity. We close the section with an example of a labeling, due to Morris [19], for which the Lemke Path for Gale Algorithm has exponential running time. This is the labeling that Savani and von Stengel [24] [25] have used to construct “hard to solve” games.

The third and last section presents our original result: a polynomial time algorithm for ANOTHER GALE, a proof 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 by Savani and von Stengel [24]. Our proof relies on a theorem by Edmonds [7] that gives a polynomial-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 correspond to the completely labeled Gale strings for the labels. We first prove the **FP** complexity of finding one of these completely labeled Gale strings. We then extend the proof to find the second string required by ANOTHER GALE.

3.1 Polynomial Parity Argument

As mentioned in section 1.3, Megiddo and Papadimitriou [17] have proved that it is impossible to have a complete problem for the class **TFNP** (*total function non-deterministic polynomial-time*) unless **NP** = **co** – **NP**, which is usually conjectured to be false. To circumvent this limitation, Papadimitriou [23] focused on the argument that proves the solution of the problems in **TFNP**, introducing, among others, the classes **PPA** (*Polynomial Parity Argument*) and **PPAD** (*Polynomial 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.” Similarly, problems in **PPAD** 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: a polynomial-sized *circuit* with n *input bits* and m *output bits* is a function $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ that can be represented with polynomially many standard “logic gates”. We define **PPAD** as the class of problems reducible to the problem END OF THE LINE, see Table 3.1. This is the definition given in Daskalakis, Goldberg and Papadimitriou [6]. The

original definition in Papadimitriou [23] is given in terms of polynomial-time Turing machines instead of the polynomial-sized circuits.

END OF THE LINE

input : Two polynomial-sized 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 construct 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 is 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. Another class relying on proofs by parity argument is **PPADS**, defined by Daskalakis, Goldberg and Papadimitriou [6]. Its definition is analogous to **PPAD**, but the output of the problem is required to be a sink of the END OF THE LINE graph. We have that **PPADS** \subseteq **PPAD** \subseteq **PPA**. It is an open problem whether the inclusion is strict.

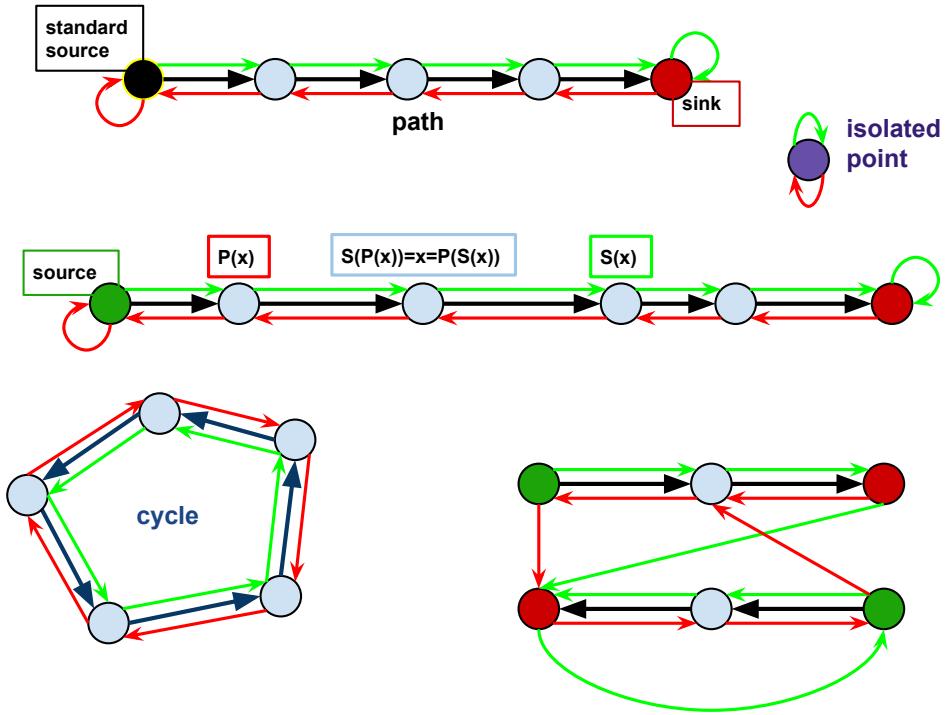


Figure 3.1 A **PPAD** problem as a directed graph with maximal indegree and outdegree 1. It consists of paths (in black), cycles (in blue) and isolated points (in purple). The input is given by the circuits S (in green) and P (in red) and the standard source (the black node). The output can be either a sink (a red node) or a nonstandard source (a green node).

As we have already noticed, the problem n -NASH, see Table 1.3, is a total function problem. Megiddo and Papadimitriou [17] have proven that it is in **TFNP**. Daskalakis, Goldberg and Papadimitriou [6] and Chen and Deng [5] 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 [6] can be found in Casetti [3].

Theorem 7. (Daskalakis, Goldberg and Papadimitriou [6]; Chen and Deng [5].) *For $n \geq 2$, the problem n -NASH is **PPAD**-complete.*

3.2 The Lemke-Howson Algorithm

Theorem 7 makes the study of solutions of n -Nash as endpoints of paths particularly interesting. In this section we will see an algorithm that describes exactly this idea.

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 P^Δ in dimension d by moving from a facet F to a facet G that shares all vertices but one with F . Since P^Δ is simplicial, there are d possible choices for G .

Suppose now that there is a labeling $l_f : [n] \rightarrow [d]$ of the facets of the simple polytope P . If we pivot from vertex x to vertex x' we “leave behind” a facet F to which x , but not x' , belong; let F have label k . At the same time, we “reach” a facet F' , to which x does not belong, but x' does; let F' have label h . Therefore, if x has labels $(l_1, \dots, k, \dots, l_d)$, then x' has labels $(l_1, \dots, h, \dots, l_d)$. We call this *dropping label k and picking up label h* , or *pivoting on label k* ; see Figure 3.2 for illustration. Analogously, if there is a labeling $l_v : [n] \rightarrow [d]$ of the vertices of the simplicial polytope P^Δ and we pivot from a facet F with labels $(l_1, \dots, k, \dots, l_d)$ to a facet F' with labels $(l_1, \dots, h, \dots, l_d)$, we say that we *drop label k and pick up label h* , or that we *pivot on label k* ; see Figure 3.3.

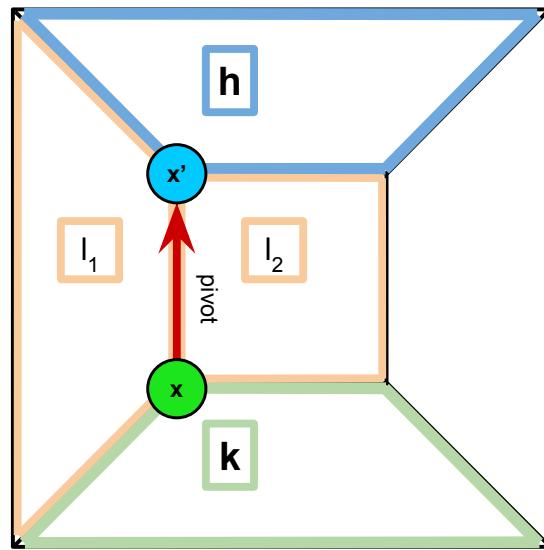


Figure 3.2 A pivot on label k , dropping vertex x with labels (l_1, l_2, k) and picking up vertex x' with labels (l_1, l_2, h) .

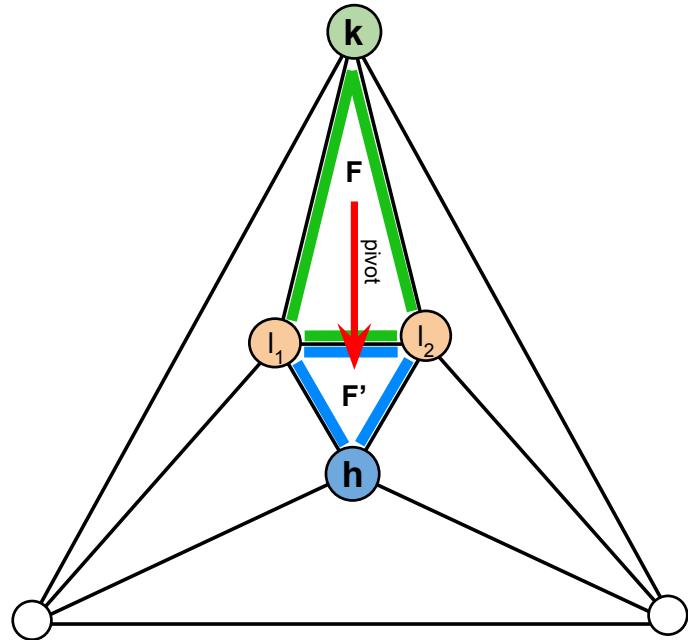


Figure 3.3 A pivot on label k , dropping facet F with labels (l_1, l_2, k) and picking up facet F' with labels (l_1, l_2, h) .

Consider a labeling function $l : [n] \rightarrow [d]$, and a subset S of $[n]$ with $|S| = d$. Then S is called *almost completely labeled* if

$$l(S) = \{l(s) \mid s \in S\} = [d] \setminus \{k\}, \quad (3.1)$$

that is, all labels appear once in S except for one *missing label* $k \in [d]$. Because $|S| = d$, in that case there is one *duplicate label* $h \in [d]$ that appears twice in S .

The set S will be the set of labels of a vertex in a simple polytope (as the set of labels of the facets containing that vertex), or the set of labels of a facet in a simplicial polytope (as the set of labels of the vertices on that facet). Correspondingly, we call such a vertex (or facet) “almost completely labeled”. It is easy to see that if we pivot from an almost completely labeled vertex (or facet) on the duplicate label, or from a completely labeled vertex (or facet) by on any label (which will become the missing label k), then we reach either an almost completely labeled or a completely labeled vertex (or facet).

The algorithm by Lemke and Howson [15] finds one Nash equilibrium of a bimatrix game. In a modern description (e.g., Savani and von Stengel [24]), it employs pivoting on the vertices of a simple polytope, moving through a succession of almost completely labeled vertices with missing label k , where this polytope is the product $P \times Q$ of the best-response polytopes. This can be abstracted slightly further by considering only a single polytope P in dimension d with facets labels from $[d]$. The resulting algorithm is also known as computing a “Lemke path”, in the terminology used by Morris [19], and shown in Algorithm 1. 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 Path Algorithm 1 returns a solution to the PPA problem ANOTHER COMPLETELY LABELED VERTEX. Furthermore, the*

Algorithm 1: Lemke Path

input : A simple d -polytope P with n facets and a labeling
 $l_f : [n] \rightarrow [d]$ of the facets of P . A vertex x_0 of P that is completely labeled by l_f .

output: A vertex $x \neq x_0$ of P that is completely labeled by l_f .

```
1 choose any label  $k \in [d]$  as missing label
2 pivot on label  $k$  from  $x_0$  to  $x$  reaching a new facet with label  $h$ 
3 while  $h \neq k$ , so  $x$  is not completely labeled do
4   pivot away from the other facet with label  $h$  from  $x$  to  $x'$ 
5   let  $h$  be the label of the new facet of  $x'$  and set  $x = x'$ 
6 return  $x$ 
```

number of completely labeled vertices in a simple polytope with labeled facets is even.

Proof. We first show that the Lemke Path Algorithm works. From the completely labeled vertex x_0 , there is a unique edge that leaves the facet with label k which leads to a new vertex x as in step 2 of the algorithm. If x is completely labeled, then the algorithm terminates with output x where clearly $x \neq x_0$. Otherwise, x is an almost completely labeled vertex with duplicate label h , where one of the facets that contain x and have label h is a “new” facet that did not contain the preceding vertex on the Lemke path. Because S is simple, x is always on exactly d facets and the duplicate label is unique. Hence no vertex (including x_0) can ever be re-visited on the path because it would otherwise offer an alternative way to proceed when the vertex was encountered for the first time.

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 start-

ing 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 starting point x_0 . The problem ANOTHER COMPLETELY LABELED VERTEX correspond to finding a non-standard endpoints of this graph, which is a **PPA** problem. \square

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 [15].

Theorem 8. (Lemke-Howson [15].) *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 $R = \{z \in \mathbb{R}^{m+n} \mid z \geq \mathbf{0}, Cz \leq \mathbf{1}\}$ be the polytope associated to the game (C, C^\top) , where

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

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 Path 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” exposition of the Lemke-Howson Algorithm by 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 R is a polytope in \mathbb{R}^{m+n} , this second version is much easier to visualize.

Example 3.1. (Savani and von Stengel [25].) Consider the 3×3 game (A, B) of Example 2.1.

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

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, which is in fact the original exposition by Shapley [26]. The path starts from $(\mathbf{0}, \mathbf{0})$. We choose the missing label 2 and move in the polytope P . Then 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 in P and y in Q , which gives here the only Nash equilibrium (x, y) of (A, B) .

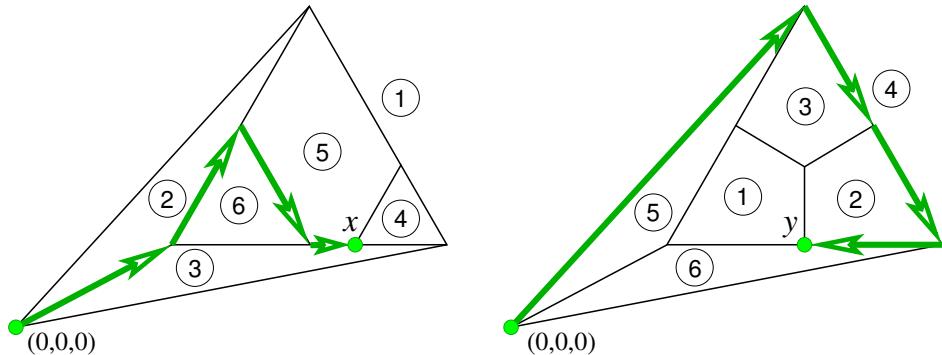


Figure 3.4 Lemke path for missing label 2 on the best response polytopes P (left) and Q (right) of game (2.2).

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. 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. (Shapley [26].) Consider the symmetric game (C, C^\top) with

$$C = \begin{pmatrix} 0 & 3 & 0 \\ 2 & 2 & 0 \\ 3 & 0 & 1 \end{pmatrix}. \quad (3.2)$$

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.

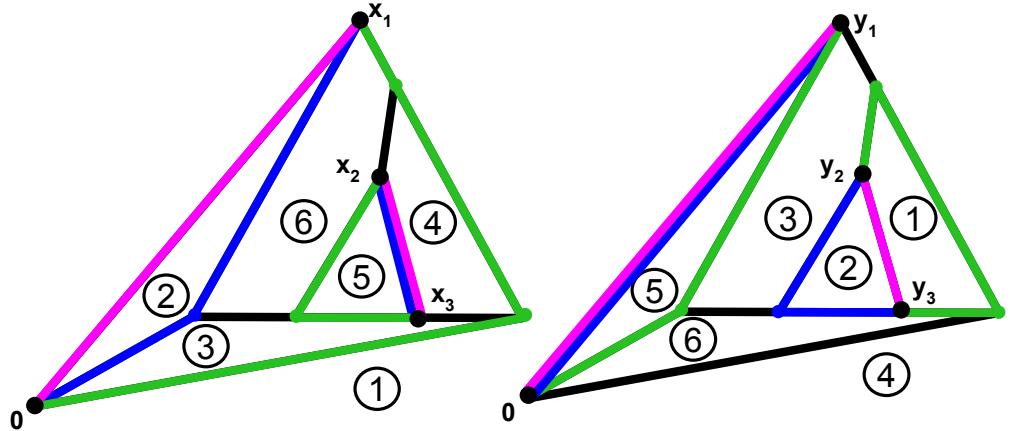


Figure 3.5 Lemke paths for missing label 1 (blue), 2 (green) and 3 (pink) on the best response polytopes of game (3.2). The paths for missing label 4, 5 and 6 on the best response polytope of player 1 and 2 are the same as the paths of 1, 2 and 3 on the best response polytope of player 2 and 1, after the corresponding change of labels.

The dual version of the Lemke Path Algorithm 1 and of Proposition 3.1 is straightforward.

Algorithm 2: Dual Lemke Path

input : A simplicial m -polytope P^Δ with n vertices and a labeling

$l_v : [n] \rightarrow [d]$ of the vertices of P^Δ . A facet F_0 of P^Δ that is completely labeled by l_v .

output: A facet $F \neq F_0$ of P^Δ that is completely labeled by l_v .

- 1 choose any label $k \in [d]$ as missing label
 - 2 pivot on label k from F_0 to F which has a new vertex with label h
 - 3 **while** $h \neq k$, so F is not completely labeled **do**
 - 4 pivot away from the other vertex with label h from F to F'
 - 5 let h be the label of the new vertex of F' and set $F = F'$
 - 6 **return** F
-

Proposition 3.2. *The Dual Lemke Path 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 Path Algorithm 1 to the polytope P^l in (2.9), or the Dual Lemke Path Algorithm (2) to the polytope P^Δ in (2.11). The following theorem by Savani and von Stengel [25] guarantees that not only does 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 $l : [n] \rightarrow [m]$. Let*

$$P = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0}, B^\top x \leq \mathbf{1}\},$$

$$Q = \{y \in \mathbb{R}^n \mid Ay \leq \mathbf{1}, y \geq \mathbf{0}\},$$

as in (2.4), and let

$$P^l = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0}, B^\top x \leq \mathbf{1}\} \quad \text{with labels in } [m] \text{ as in (2.10)}$$

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 Gale games. We will consider the strings $s \in G(d, n)$, with d even, as “wrapped-around strings”. Let $s(i) = 1$ for an index $i \in [n]$. Then, by the Gale evenness condition, there is an odd run of $\mathbf{1}$ ’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 from s to s'* is given by setting $s'(i) = 0$ and $s'(j) = 1$ to yield the new string s' which otherwise agrees with s . If there is a labeling $l_s : [n] \rightarrow [d]$, we say that we *pivot on label $l_s(i)$* (where we have to specify i when the label $l_s(i)$ does not uniquely identify i), *dropping label $l_s(i)$* and *picking up label $l_s(j)$* . The *Lemke Path for Gale Algorithm* is given in Algorithm 3.

Algorithm 3: Lemke Path for Gale

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

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

```
1 choose a missing label  $k \in [d]$ 
2 pivot on label  $k$  from  $s_0$  to  $s$  reaching a new 1 bit with label  $h$ 
3 while  $h \neq k$ , so  $s$  is not completely labeled do
4   pivot away from the other 1 bit in  $s$  with label  $h$  from  $s$  to  $s'$ 
5   let  $h$  be the label of the new 1 bit in  $s'$  and set  $s = s'$ 
6 return  $s$ 
```

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

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

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

This corresponds to the labeling l_s for $G(4, 6)$ given in Example 2.13, which gives the four completely labeled Gale strings $s_A = \mathbf{111100}$, $s_B = \mathbf{110110}$, $s_C = \mathbf{100111}$ and $s_D = \mathbf{101101}$. These correspond to the facets A , B , C and D of $C_4(6)$. Pivoting from $s_A = \mathbf{111100}$ with missing label 3 returns $s_B = \mathbf{110110}$. Analogously, pivoting from facet A with missing label 3 returns facet B .

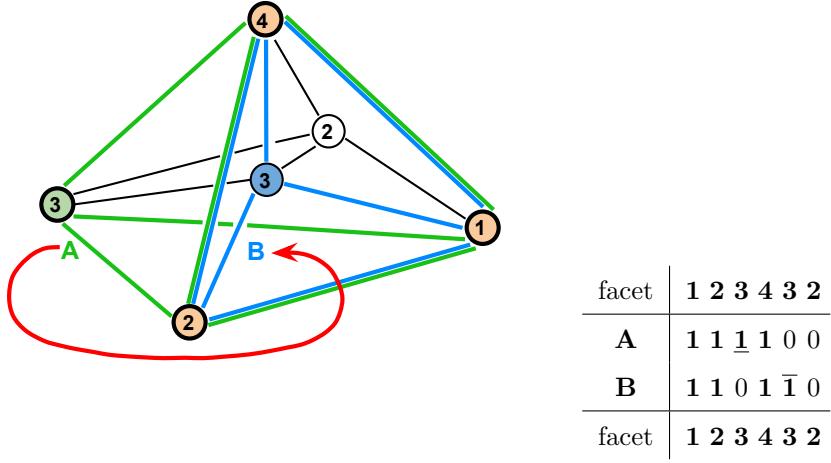


Figure 3.6 The pivoting to $s_A = \mathbf{111100}$ to $s_B = \mathbf{110110}$ in the Lemke Path for Gale Algorithm corresponds to the pivoting from facet A to facet B in the Dual Lemke Path Algorithm.

The membership of ANOTHER GALE in the complexity class **PPA** follows from an argument similar to Proposition 3.1. We extend the result to prove its membership in **PPAD**.

A *permutation* of elements of an ordered set S is a sequence without repetition. This gives a rearrangement of the elements of S . A *transposition* is a permutation of exactly two elements. The *sign of a permutation* is $\text{sign}(\sigma) = (-1)^m$, where m is the number of transpositions needed to get the *natural order* $\sigma_0 = 1 \dots n$ from σ . It is immediate to see that any two permutations that differ in only one transposition have opposite sign. We define the *sign of a completely labeled Gale string* $s \in G(d, n)$ as follows: let $l : [n] \rightarrow [d]$ be the labeling of $G(d, n)$, and let l_0 be the string of labels $l(i)$ such that $s(i) = \mathbf{1}$ 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 $\mathbf{1}^d 0^{(n-d)}$ 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 $s(i_2) = \mathbf{1}$ 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 Path for Gale Algorithm in the case where $\text{sign}(s_0) = +1$. The negative case is analogous, 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 $\text{sign}(l_1) = -1$, which implies $\text{sign}(l_2) = +1$. The next pivoting step drops the label h from index i_2 , so again we change sign. This shows that the Lemke Path for Gale Algorithm results in the sign of the completely and almost 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.

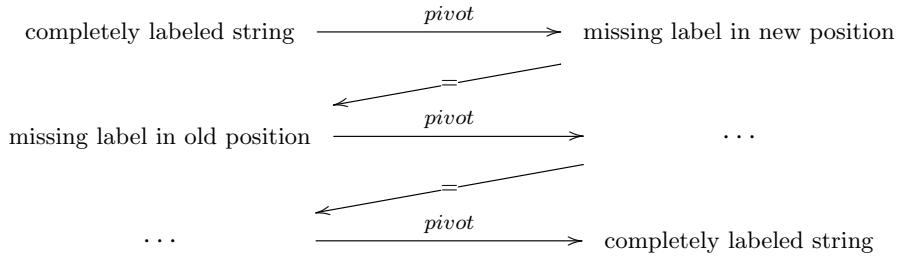


Table 3.2 Sign switching on the Lemke Path for Gale Algorithm.

Proposition 3.3. *The Lemke Path for Gale Algorithm 3 returns a solution to the PPAD problem ANOTHER GALE. Furthermore, the number of completely*

labeled Gale strings $s \in G(d, n)$ is even, and the two completely labeled Gale strings at opposite ends of any Lemke path have opposite sign.

Example 3.4. Let $l_s = 123432$. Consider the Lemke path from the completely labeled Gale string $s = 1\mathbf{1}\mathbf{1}100$ with missing label 1. Figure 3.7 shows the graph of Table 3.2. Notice that $\text{sign}(101101) = \text{sign}(l(6)l(1)l(3)l(4)) = \text{sign}((2134))$, since $s(6) = s(1) = 1$ and therefore the indices 6 and 1 are consecutive in the same run.

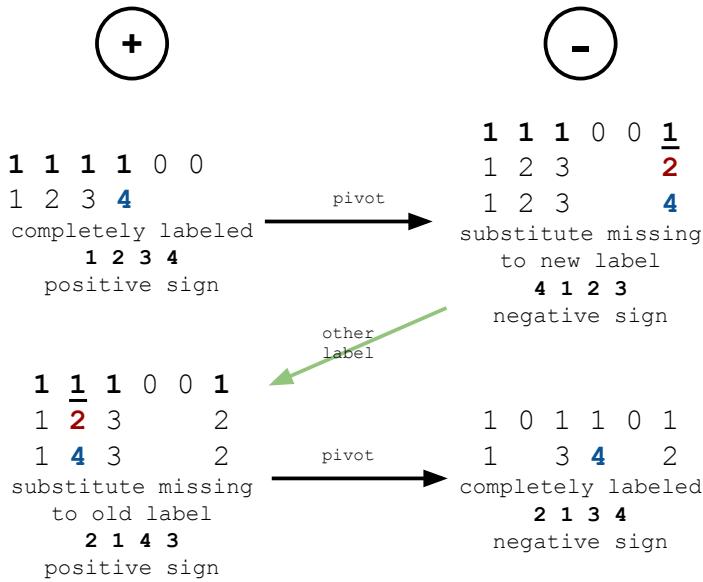


Figure 3.7 Pivoting with sign on 123432.

A generalization of Proposition 3.3 is given in Shapley [26]: two Nash equilibria at the ends of a Lemke path have opposite *index*, a concept analogous to the sign but defined using determinants on the payoff matrices for the equilibrium support. The index of a Nash equilibrium is usually normalized (by multiplication with -1 of all signs if necessary) so that the artificial equilibrium has index -1 . Then a nondegenerate game with n Nash equilibria with index $+1$ has $n - 1$ Nash equilibria with index -1 .

Morris [19] gave an example of a labeling $l : [2d] \rightarrow [d]$ where the length of the Lemke paths on the cyclic polytope $C_d(2d)$ for the Lemke Path Algorithm 1 grows exponentially in d for every missing label. These paths are therefore also exponential on $G(d, 2d)$ for the Lemke Path for Gale Algorithm 3. The labeling by Morris is given for d both even and odd as follows:

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

Without repetitions of labels in consecutive positions, and restricted to the case of even d , Morris's labeling is simplified to $l : [2d-2] \rightarrow [d]$ as follows:

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

Savani and von Stengel [24] [25] extend Morris's example in order to construct different “hard to solve” games. These results give a strong motivation to study the complexity 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 Path for Gale Algorithm by using a different algorithm.

Example 3.5. Consider the labeling $l = 1234564523$ for $G(6, 10)$. The only two completely labeled Gale string are $s = \underline{\mathbf{1}}\mathbf{1}\mathbf{1}\mathbf{1}\mathbf{1}00$ and $s' = \mathbf{1}00000\underline{\mathbf{1}}\mathbf{1}\mathbf{1}\mathbf{1}$. Table 3.3 shows the Lemke path for missing label 1.

1 2 3 4 5 6 4 5 2 3
<hr/>
<u>1</u> 1 1 1 1 1 0 0 0 0
0 1 1 <u>1</u> 1 1 1 0 0 0
0 1 1 0 <u>1</u> 1 1 1 0 0
0 <u>1</u> 1 0 0 1 1 1 1 0
0 0 1 1 0 1 <u>1</u> 1 1 0
0 0 1 1 1 1 0 <u>1</u> 1 0
0 0 <u>1</u> 1 1 1 0 0 1 1
0 0 0 <u>1</u> 1 1 1 0 1 1
1 0 0 0 0 1 1 1 1 1
<hr/>
1 2 3 4 5 6 4 5 2 3

Table 3.3 The Lemke path for the Morris labeling on $G(4, 6)$ with missing label 1. The bit position **1** that is dropped in the next pivoting step is underlined, the bit **1** that has just been picked up is overlined.

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 [7] proved the **FP** complexity of finding a perfect matching. The result can be easily extended to multigraphs which have parallel edges, since a

perfect matching of a multigraph G corresponds to a perfect matching of the graph G' obtained by ignoring all parallel edges of G .

Theorem 10. (Edmonds [7].) *It takes polynomial time to find a perfect matching of a graph or to decide that no such matching exists.*

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 3.4 The problem PERFECT MATCHING.

Proposition 3.4. *PERFECT MATCHING is in FP.*

To prove our main result on ANOTHER GALE, we will prove that the related problem GALE can be solved in polynomial time.

GALE

input : A labeling $l : [n] \rightarrow [d]$, where d is even and $n > d$.

output: A Gale string $s \in G(d, n)$ that is completely labeled by l or No if no such string exists.

Table 3.5 The problem GALE.

Theorem 11. *GALE is in FP.*

Proof. Recall that we consider the Gale strings $s \in G(d, n)$ as “wrapped-around strings”, that is, $s(n + i) = s(i)$ and $l(n + i) = l(i)$ for all $i \in [n]$.

Let $G = (V, E)$ be the multigraph with $V = [d]$, so that the vertices of G correspond to the labels $l(i) \in [d]$, and

$$E = \{ (l(i), l(i + 1)) \mid i \in [n] \}, \quad (3.3)$$

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 for l . By the Gale evenness condition, every run of length k in s corresponds uniquely to $k/2$ disjoint pairs of indices $(i, i + 1)$ with $s(i) = s(i + 1) = 1$. Let M be the set of corresponding edges $(l(i), l(i + 1))$ in E . Since s is completely labeled, every label $l(i) \in [d]$ occurs at exactly one of the endpoints of an edge in M . Since there are no repetitions of i 's in M , there is only one such edge. Hence, M is a perfect matching of G .

Conversely, let M be a perfect matching for $G = (V, E)$. Let s be the 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, every vertex $l(i) \in [d]$ is the endpoint of an edge $(l(i), l(i + 1)) \in M$, therefore s is completely labeled.

This proves that GALE reduces to PERFECT MATCHING in polynomial time. By Proposition 3.4, Algorithm 4 takes polynomial time to find a solution of an instance of GALE or to decide that no such solution exists. \square

It is interesting to notice that the graph $G = (V, E)$ is an Euler graph, since its edges have been defined following an Euler tour starting in $l(1)$.

Algorithm 4: Find a Completely Labeled Gale String

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

output: A Gale string $s \in G(d, n)$ that is completely labeled by l or
No if no such string exists.

- 1 Set $G = (V, E)$, with $V = [d]$ and $E = \emptyset$
- 2 **for** $i \in [n - 1]$ **do**
- 3 $E = E \cup \{(l(i), l(i + 1))\}$
- 4 $E = E \cup \{(l(n), l(1))\}$
- 5 Run Edmonds's Algorithm on G
- 6 **if** *Edmonds's Algorithm returns a perfect matching M of G* **then**
- 7 set $s = 0^n$
- 8 **for** $(l(i), l(j)) \in M$ **do**
- 9 set $s(i) = s(j) = 1$
- 10 Return s
- 11 **else**
- 12 Return No

Example 3.6. Figure 3.8 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 $s = \mathbf{1111110000}$ and $s = \mathbf{1000011111}$.

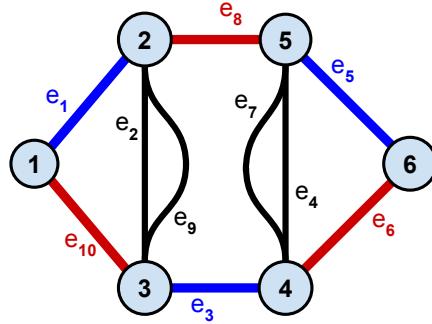


Figure 3.8 The perfect matchings for the Morris labeling $l = 1234564523$, corresponding to the Gale strings $s = \mathbf{1111110000}$ (in blue) and $s = \mathbf{1000011111}$ (in red).

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

Example 3.7. Consider the labeling $l_s = 121314$. Figure 3.9 shows the corresponding graph. It is easy to see that a perfect matching is not possible. Analogously, as we have seen in Example 2.12, there isn't any possible completely labeled Gale string for l_s .

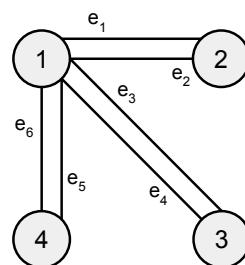


Figure 3.9 The graph for the labeling $l = 121314$

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

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

Proof. Let $G = (V, E)$ be the graph corresponding to the labeling $l : [n] \rightarrow [d]$ with $V = [d]$ and E as in (3.3). 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 a 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 $E(G_i) \subset E(G)$, a perfect matching M_i of G_i is a perfect matching for G as well. We have that $e_i \notin M_i \subset E_i$ but $e_i \in M_0$. Therefore $M_i \neq M_0$.

There are exactly $d/2$ possible G_i 's. By Proposition 10, finding a perfect matching or deciding that there is none takes polynomial time for each G_i . Therefore, by applying Algorithm (5) it is possible to find a perfect matching $M_i \neq M_0$ and the corresponding completely labeled Gale string $s \neq s_0$ in polynomial time. \square

Applying Theorem 2.3 to Theorem 12 we have our main result: GALE NASH is in **FP**, under the assumption that the construction of the game from a cyclic polytope is given.

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

Algorithm 5: Find Another Gale String

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

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

```
1 set  $G = (V, E)$  as in Algorithm 4
2 set  $M_0$  as the matching corresponding to  $s_0$ 
3 for  $m \in M_0$  do
4    $E_m = E \setminus \{m\}$ 
5   Run Edmonds's Algorithm on  $G_m = (V, E_m)$ 
6   if Edmonds's Algorithm returns a perfect matching  $M_m$  of  $G_m$ 
7     then
8       set  $s = 0^n$ 
9       for  $(l(i), l(j)) \in M_m$  do
10         set  $s(i) = s(j) = 1$ 
11   return  $s$ 
```

Example 3.8. The labeling $l = 123432$ gives the graph G in Figure 3.10. Suppose that Edmonds's algorithm returns the matching $M_0 = \{e_1, e_3\}$ of Figure 3.10 (left), corresponding to the completely labeled Gale string $s_0 = \mathbf{111100}$. 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 3.10 (right).

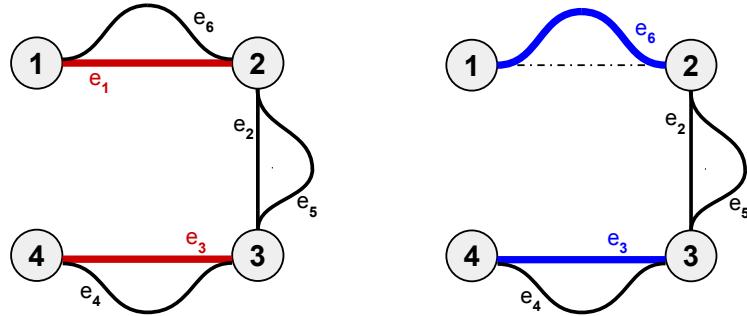


Figure 3.10 Left: The graph for the labeling $l = 123432$ with the perfect matchings corresponding to the completely labeled Gale strings $s_0 = \mathbf{111100}$. Right: The graph $G_1 = (V(G), E(G) \setminus \{e_1\})$ and the perfect matching corresponding to the Gale string $s = \mathbf{101101}$.

Example 3.9. Consider the Morris graph of Example 3.6. Suppose that Edmonds's algorithm returns the perfect matching $M_0 = \{e_1, e_3, e_5\}$, as in Figure 3.11 (left), corresponding to the completely labeled Gale string $s_0 = \mathbf{1111110000}$. We can then delete the edge e_1 to obtain the graph G_1 , as in Figure 3.11 (center). The graph G_1 has a perfect matching $M = \{e_6, e_8, e_{10}\}$, shown in Figure 3.11 (right). This is also a perfect matching of G . It corresponds to the only other completely labeled Gale string $s = \mathbf{1000011111}$.

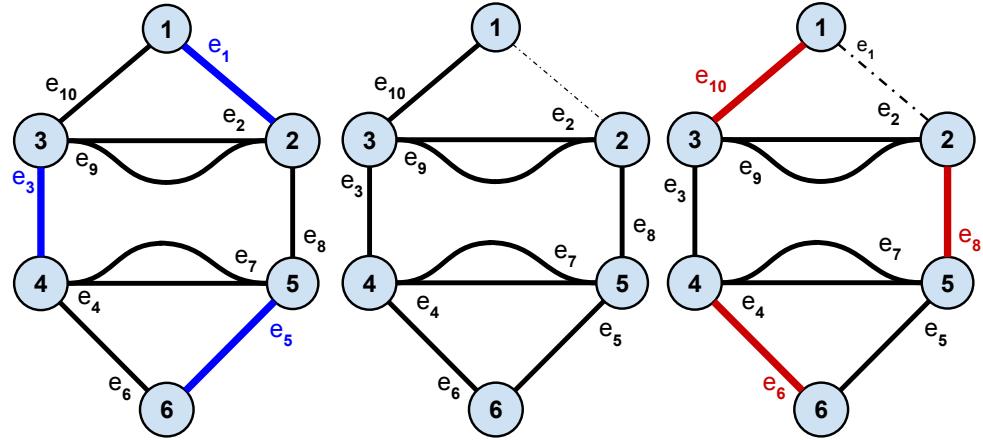


Figure 3.11 Left: The Morris graph $G = (V, E)$ and its matching $M = \{e_1, e_3, e_5\}$. Center: The graph $G_1 = (V(G), E(G) \setminus \{e_1\})$. Right: The set $M_1 = \{e_6, e_8, e_{10}\}$ is a perfect matching of both G_1 and G .

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 does not 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 **PPADS** problem OPPOSITE SIGN GALE of Table 4.1.

OPPOSITE SIGN GALE

input : A labeling $l : [n] \rightarrow [d]$ 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 polynomial-time algorithm for OPPOSITE SIGN GALE where the labeling for which the Gale graph $G = (V, E)$ in (3.3) is planar has been given by Merschen [18]. The result for any labeling has been given by Végh and von

Stengel [27]. The latter uses the definition of a general framework to deal with pivoting algorithms, called *Complementary Pivoting with Direction Algorithm*. This is defined not only for perfect matchings of a Gale graph, but for the wider class of room partitions of Euler complexes, first introduced by Edmonds [8] and Edmonds and Sanità [9].

A *d-dimensional Euler complex* (in the following: *d-oik*) is $\mathcal{C} = (V, R)$, where V is a finite set of *nodes* and R is a family of subsets R_i of V , called *rooms*, such that $|R_i| = d$ for all i and any set of $d - 1$ nodes (called a *wall*) is contained in an *even* number of rooms. A *room partitioning* M of (V, R) is a subset M of R such that each node $v \in V$ is in exactly one room $R_i \in M$. Edmonds and Sanità used an “Exchange Algorithm” and a parity argument analogous to Proposition 3.1 to show that there is an even number of room partitions of an oik. In the case where the rooms in the oik are defined as the sets of facets incident to the vertices of the best response polytopes of a bimatrix game and the room partitions correspond to completely labeled vertex pairs, the Lemke-Howson Algorithm is a special case of the Exchange Algorithm. An simpler example of an oik is an Euler graph, with vertices as nodes of the graph, rooms as the edges of the Euler graph, and the perfect matchings as room partitions. Furthermore, the Gale graph is an Euler graph. This points towards a further connection between oiks and the topic of our study. This connection is less trivial than it appears, as it can be seen as soon as we look into the issue of giving a sign to room partitions and, by extension, to Nash equilibria.

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

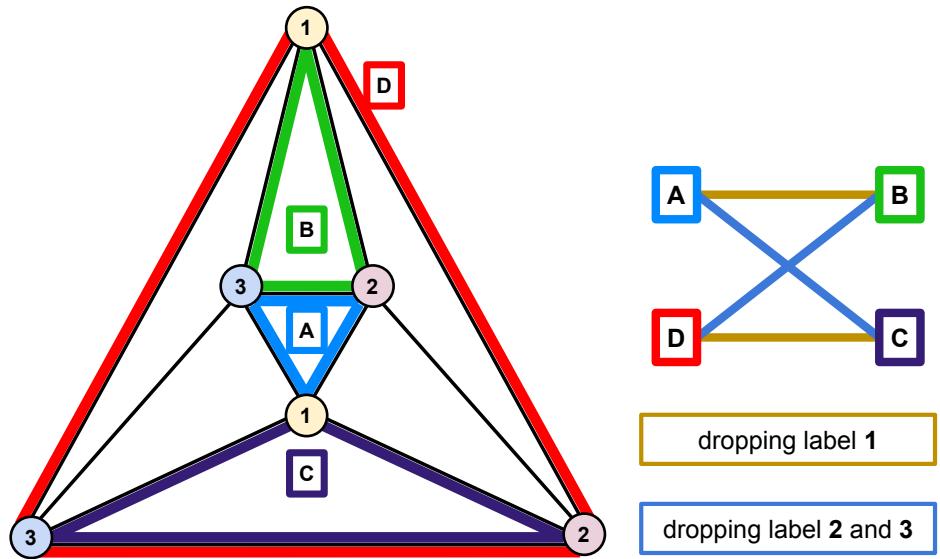


Figure 4.1 Left: A labeled octahedron and its completely labeled facets.

Right: The endpoints of the Lemke paths on the octahedron.

On the other hand, consider the same octahedron from the point of view of room partitions. Figure 4.2 shows the Exchange Algorithm of Edmonds and Sanità [9] from room partition B dropping vertex $v = 1$.

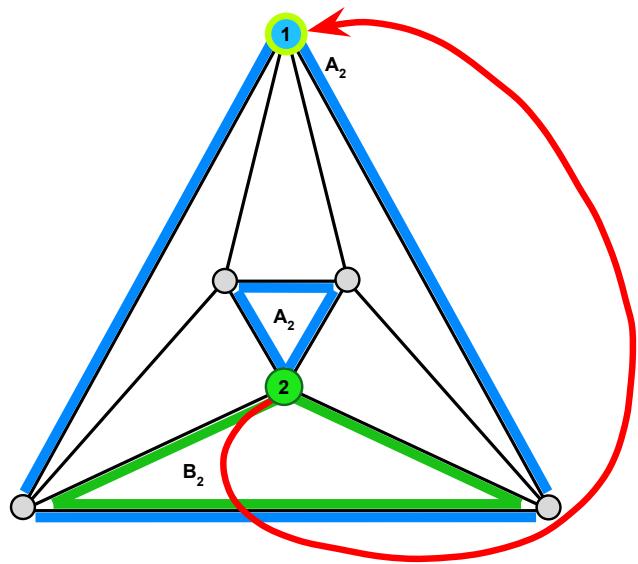
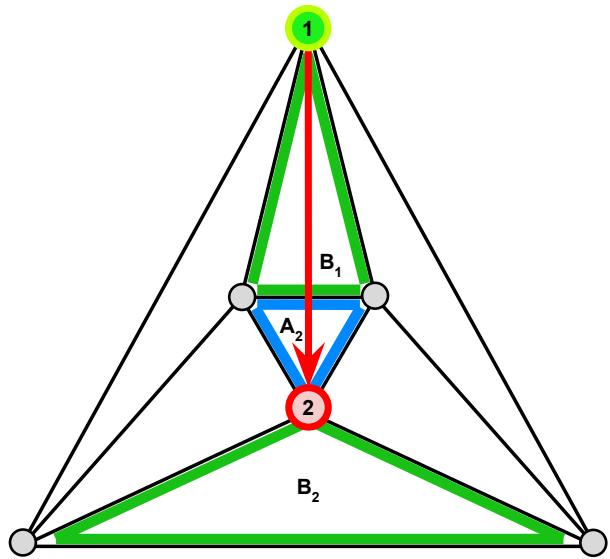


Figure 4.2 Top: Pivot on vertex 1 from room B_1 to room A_1 . The new vertex 4 is duplicate in room A_1 and room B_2 . Bottom: Pivot on vertex 4 from the old room B_2 to room A_2 , picking up the missing vertex.

The graph describing the endpoints of the Exchange Algorithm, analogous to Figure 4.1, is shown in Figure 4.3 and it is not bipartite. Giving a sign to the room partitions cannot be done simply through Edmonds's Exchange Algorithm.

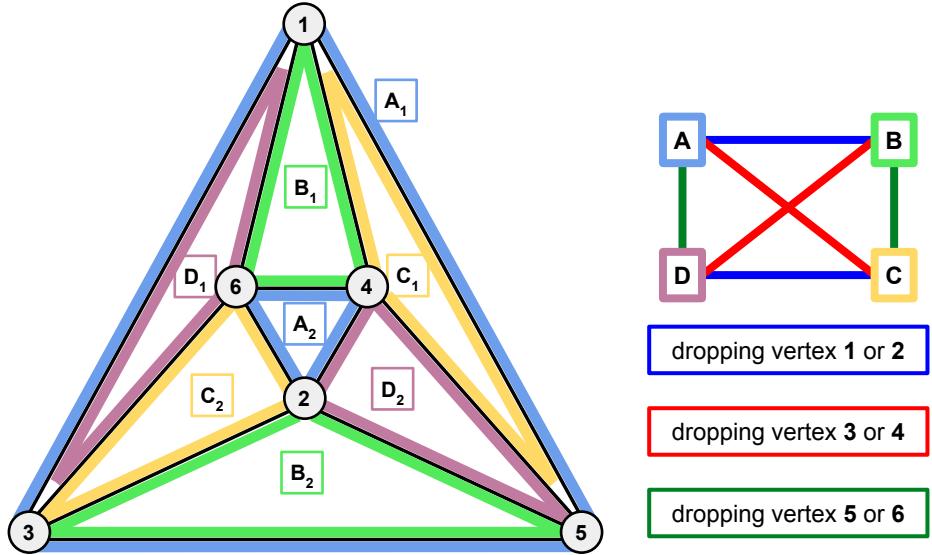


Figure 4.3 Left: The room partitions of an octahedron.

Right: The endpoints of Exchange Algorithm on the octahedron.

Végh and von Stengel [27] 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, where the order of the rooms in the partition matters if the dimension d of the oik matters if d is odd. If d is even, as in the case of perfect matchings of an Euler graph, the order of the rooms does not matter. The parity result follows similar to the previous cases. 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 [27] 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 [14] as **PSPACE**-complete. This invites, once more, a further investigation of exchange-like algorithms to construct “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 based 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 [24] was circumvented together with the simpler case of Gale games by our result.

Another result worth mentioning is found in Merschen [18]: finding the number of equilibria of a Gale game is **#P**-complete. Gilboa and Zemel [12] 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 [30] 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 open borderline case for the complexity of games in a more general framework.

Acknowledgments

But I... We are so much else. I feel an essay coming on. Cassandra Igarashi, [13]

This thesis comes at the end of a long journey in which I have been supported by more people than I can mention. So, first of all: apologies to everyone who I will forget to thank for their help in nine years as a student at LSE.

Someone I could never forget to thank is my supervisor Bernhard von Stengel. His support, his trust, and his patience have been incredible; the ideas that I studied under his guidance have become a treasured part of myself.

Discussions with my co-author Julian Merschen have always been an inexhaustible source of new ideas and challenges. Of all the professors in the Mathematics department, special thanks must go to Laszlo Végh for stimulating conversations, and to Graham Brightwell and Jan van den Heuvel for their incredible lectures and their kindness. Anne, Ahmed, Hessah and Stephan have made my life better with more than chatting while the coffee was brewing, and even that was a pleasure.

I had the honour to meet many of the people mentioned in the bibliography of this thesis. I must mention Jack Edmonds at Hammamet; Paul Goldberg and Rahul Savani from Warwick to anywhere; Constantinos Daskalakis, Xiaotie Deng, Nimrod Megiddo and Walter Morris in Dagstuhl; Xi Chen at LSE; brushing shoulders with John Nash in São Paulo. Christos Papadimitriou deserves a special note: his incredible human side made his intellectual one shine even brighter.

I wouldn't have survived these years without the support services at LSE. Without Dave, Jackie and Rebecca I wouldn't have seen so many opportunities that the Maths department offered me, from the reminder of seminars to the help with financial support. Loraine, Louisa and everyone at the Research Degrees Unit put up with my delays of delays of... (it's not a typo, it's recursion). Jane Sedgwick and Sue Haines at the Disability Office have literally saved my life. Viki Chinn at the Career Centre pushed me to do – and I did.

There is life outside mathematics, despite the tales of PhD Comics, and some of it is wonderful. Cristiana has been giving me comfort and tea day and night, no matter how desperate I was. Paolo has been a light in dark times, and his “cicoria” are words to live by. Violetta has given a space to my writing, and put up with too many silences. Alessandro is more than the Avvocato Laser, and Stefano has given me hope for the future.

I wouldn't have known how beautiful theorems can be without my mother, and I would probably know more about giving up without my father.

A lifetime will not be enough to thank my husband Emmanuele; I'll keep it short here and have a longer dinner.

Bibliography

- [1] A. V. Balthasar (2009). “Geometry and equilibria in bimatrix games.” PhD Thesis, London School of Economics and Political Science.
- [2] G. R. Brightwell, P. Winkler (2004). “Note on Counting Eulerian Circuits.” CDAM Research Report LSE-CDAM-2004-12.
- [3] M. M. Casetti (2008). “PPAD Completeness of Equilibrium Computation.” MSc Thesis, London School of Economics and Political Science.
- [4] M. M. Casetti, J. Merschen, B. von Stengel (2010). “Finding Gale Strings.” *Electronic Notes in Discrete Mathematics* 36, pp. 1065–1082.
- [5] X. Chen, X. Deng (2006). “Settling the Complexity of 2-Player Nash Equilibrium.” *Proc. 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 261–272.
- [6] C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou (2009). “The Complexity of Computing a Nash Equilibrium.” *SIAM Journal on Computing* 39, pp. 195–259.
- [7] J. Edmonds (1965). “Paths, Trees, and Flowers.” *Canad. J. Math.* 17, pp. 449–467.
- [8] J. Edmonds (2009). “Euler complexes.” In: *Research Trends in Combinatorial Optimization*, eds. W. Cook, L. Lovasz, and J. Vygen, Springer, Berlin, pp. 65–68.
- [9] J. Edmonds, L. Sanità (2010). “On finding another room-partitioning of the vertices.” *Electronic Notes in Discrete Mathematics* 36, pp. 1257–1264.

- [10] D. Gale (1963). “Neighborly and Cyclic Polytopes.” In: *Convexity, Proc. Symposia in Pure Math.*, Vol. 7, ed. V. Klee, American Math. Soc., Providence, Rhode Island, pp. 225–232.
- [11] D. Gale, H. W. Kuhn, A. W. Tucker (1950). “On Symmetric Games.” In: *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.
- [12] I. Gilboa, E. Zemel (1989). “Nash and correlated equilibria: some complexity considerations.” *Games and Economic Behavior* 1, pp. 80–93.
- [13] K. Gillen, J. McKelvie, M. Wilson (2015). “Fear and Loathing in Eternity.” *The Wicked + The Divine*, issue 9, ed. Image Comics.
- [14] P. W. Goldberg, C. H. Papadimitriou, R. Savani (2011). “The Complexity of the Homotopy Method, Equilibrium Selection, and Lemke-Howson solutions.” *Proc. 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 67–76.
- [15] C. E. Lemke, J. T. Howson, Jr. (1964). “Equilibrium Points of Bimatrix Games.” *J. Soc. Indust. Appl. Mathematics* 12, pp. 413–423.
- [16] A. McLennan, R. Tourky (2010). “Imitation Games and Computation.” *Games and Economic Behavior* 70, pp. 4–11.
- [17] N. Megiddo, C. H. Papadimitriou (1991). “On Total Functions, Existence Theorems and Computational Complexity.” *Theoretical Computer Science* 81, pp. 317–324.
- [18] J. Merschen (2012). “Nash Equilibria, Gale Strings, and Perfect Matchings.” PhD Thesis, London School of Economics and Political Science.
- [19] W. D. Morris Jr. (1994). “Lemke Paths on Simple Polytopes.” *Math. Oper. Res.* 19, pp. 780–789.
- [20] J. F. Nash (1951). “Noncooperative games.” *Annals of Mathematics*, 54, pp. 289–295.
- [21] M. J. Osborne, A. Rubinstein (1994). *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts.

- [22] C. H. Papadimitriou (1994). *Computational Complexity*. Addison-Wesley, Reading, MA.
- [23] 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.
- [24] R. Savani, B. von Stengel (2006). “Hard-to-solve Bimatrix Games.” *Econometrica* 74, pp. 397–429.
- [25] R. Savani, B. von Stengel (2015). “Unit Vector Games.” arXiv:1501.02243v1 [cs.GT]
- [26] L. S. Shapley (1974). “A Note on the Lemke-Howson Algorithm.” *Mathematical Programming Study 1: Pivoting and Extensions*, pp. 175–189
- [27] L. A. Végh, B. von Stengel (2015), “Oriented Euler Complexes and Signed Perfect Matchings.” *Mathematical Programming Series B* 150, pp. 153–178.
- [28] J. von Neumann (1928). “Zur Theorie der Gesellschaftsspiele.” *Mathematische Annalen* 100, pp. 295–320.
- [29] 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.
- [30] B. von Stengel (2012). “Completely Labeled Facet is NP-Complete.” Manuscript, 6 pp.
- [31] G. M. Ziegler (1995). *Lectures on Polytopes*. Springer, New York.