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, April 2015

Declaration

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

Abstract

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

These games were used by Savani and von Stengel [19] as an example of exponential running time for the classical Lemke-Howson algorithm to find a Nash equilibrium of a bimatrix game [11]. It was conjectured that solving these games via the Lemke-Howson algorithm was complete in the class **PPAD** (Proof by Parity Argument, Directed version). A major motivation for the definition of this class by Papadimitriou [18] was, in turn, to capture the pivoting technique of many results related to the Nash equilibrium, including the Lemke-Howson algorithm.

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

Our proof exploits two results. The first one is the representation of the Nash equilibria of these games as Gale strings, as seen in Savani and von Stengel [19]. The second one is the polynomial-time solvability of the problem of finding a perfect matching in a graph, proven by Edmonds [6].

Merschen [14] and Végh and von Stengel [22] expanded our technique to prove further interesting results about index of a Nash equilibrium (see Shapley [21]) and orientation of an oik (see Edmonds [7] and Edmonds and Sanità [8]).

Contents

List of Figures

Introduction

The topic of this thesis is a problem in the field of *algorithmic game theory*, that is, the study of game-theoretic problems from the point of view of computer science. In particular, we focus on the computational complexity of a particular class of games. These

General refs for comp compl [17]

General refs for geometry [26]

from SvS15

Savani and von Stengel (2006) showed that the LH algorithm may take exponentially many steps. Their construction uses âĂIJdual cyclic polytopesâĂİ which have a well-known vertex structure for any dimension and number of linear inequalities. Morris (1994) used similarly labeled dual cyclic polytopes where all âĂIJLemke pathsâĂİ are exponentially long. A Lemke path is related to the path computed by the LH algorithm, but is defined on a single polytope that does not have a product structure corresponding to a bimatrix game. The completely labeled vertex found by a Lemke path can be interpreted as a symmetric equilibrium of a symmetric bimatrix game. However, as in the example in Figure 4 below, such a symmetric game may also have nonsymmetric equilibria which here are easy to compute, so that the result by Morris (1994) seemed not suitable to describe games that are hard to solve with the LH algorithm.

The âĂIJimitiation gamesâĂİ defined by McLennan and Tourky (2010) changed this picture. In an imitation game, the payoff matrix of one of the players is the identity matrix. The mixed strategy of that player in any Nash equilibrium of the imitation game corresponds exactly to a symmetric equilibrium of the symmetric game defined by the payoff matrix of the other player. In that way, an algorithm that finds a Nash equilibrium of a bimatrix game can be used to find a symmetric Nash equilibrium of a symmetric game.

In one sense the two-polytope construction of Savani and von Stengel (2006) was overly com- plicated: the imitation games by McLennan and Tourky (2010) provide a simple and elegant way to turn the single-polytope construction of Morris (1994) into exponentially-long LH paths for bimatrix games. In another sense, the construction of Savani and von Stengel was not redun- dant, since it provided examples that are simultaneously bad for the LH algorithm and âĂIJsupport enumerationâĂİ, which is another natural and simple algorithm for finding equilibria. The support of a mixed strategy is the set of pure strategies that are played with positive probability. Given a pair of supports of equal size, the mixed strategy probabilities are found by equating all payoffs for the other playerâĂŹs support, which then have to be compared with payoffs outside the support to establish the equilibrium property (see Dickhaut and Kaplan, 1991).

In this paper, we extend the idea of imitation games to games where one payoff matrix is arbi- trary and the other is a set of unit vectors. We call these unit vector games.

we use them to extend MorrisâĂŹs con- struction to give bimatrix games that use only one dual cyclic polytope, rather than the two used by Savani and von Stengel, and for which both the LH algorithm and support enumeration are simultaneously bad. This result (Theorem 11) was first described by Savani (2006, Section 3.8).

summary of chapt 2:

We begin this section with the definition of almost complete labeling; we then move on to the classic version of the Lemke-Howson algorithm for the problem Another Completely Labeled Vertex, as given in the beautiful exposition by Shapley [21], and its dual version for Another Completely Labeled Facet. Finally, we present the Lemke-Howson for Gale algorithm. In the next session we will tackle the issue of the computational complexity of these algorithms: Another Completely Labeled Facet and Another Completely Labeled Vertex are **PPA**, Nash is **PPAD**, as first shown in Papadimitriou [18]; furhermore, as shown by Morris [15] and by Savani and von Stengel [19], there are cases of exponential running time. This had led us to conjecture that these problems could be exploited for a proof of **PPAD** completeness, also considering that finding a completely labeled facet (or vertex, or the existence of a Nash equilibrium) is **NP** in the case of a general labeled polytope, as proven by von Stengel [25] . In the last section we will finally present our original result, that goes in the opposite direction: the problem Another GALEcan be solved in polynomial time, that is, it is a problem in **TFP**. Unit vector games with dual cyclic best response polytope present therefore a case apart, as expected, but not because they are harder than others, but because they are easier.

check proof, citation

original? main?

Chapter 1

Preliminary definitions

We begin by giving some background definitions and notation that will be used throughout this thesis. The first section will cover polytopes (see Ziegler [26] for further details); the second section will deal with basic game theory. We will finally focus on computational complexity; after the standard definitions (see Papadimitriou [17] for an introduction) we will move on to the more recently defined classes **TFNP** and **PPAD**, first introduced in [13] and [18], respectively. The latter, in particular, is a key concept in the study of the problems that are the focus of this thesis.

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

1.1 Vectors and Polytopes

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

An *affine combination* of points in an Euclidean space $\{z_1, \ldots, z_m\} \subset^n$ 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 $z_1, \ldots, z_m$ are *affinely independent* if none of them is an affine combination of the others. The *convex hull* of the points $z_1, \ldots, z_m$ is an affine combination with $\lambda_i \geq 0$ for all $i \in [n]$; we denote it as $\mathrm{conv}\{z_1, \ldots, z_m\} = \{\sum_i \lambda_i z_i \mid \lambda_i \geq 0, \; \sum_i \lambda_i = 1\}$. A set of points $Z = \{z_1, \ldots, z_m\}$ is *convex* if $Z = \mathrm{conv}(Z)$, and it has *dimension* d if Z has exactly $d+1$ affinely independent points. A convex set of dimension d is called a *d-simplex*. The *standard d-simplex* is $\Delta_d = \mathrm{conv}\{e_1, \ldots, e_{d+1}\}$.

A *polytope* is the convex hull of a finite set of points $\{z_1, \ldots, z_m\} \subset \mathbb{R}^n$, not necessarily affinely independent; the *dimension* of the polytope is the dimension of its convex hull. A polyhedron is the intersection of finitely many closed halfspaces $\{x \in \mathbb{R}^d \mid a^T x \leq a_0\}$; note that a bounded polyhedron is a polytope. A *vertex* of a d-dimensional polytope $P = \mathrm{conv}(Z)$ is a point $z \in Z$ such that $\mathrm{conv}(Z \setminus \{z\}) \neq P$; an *edge* of P is a 1-dimensional line segment that has two vertices as endpoints. A *facet* of P is the convex hull of a set of d vertices $F = \{z_1, \ldots, z_d\}$ that lie on a hyperplane of the form $\{x \in \mathbb{R}^d \mid a^T x = a_0\}$ so that $a^T u < a_0$ for all other vertices u of P; the vector a (taken as unique up to a scalar multiple) is called the *normal vector* of the facet.

A *d-dimensional simplicial polytope* P is the convex hull of a set of at least $d+1$ points $v \in \mathbb{R}^d$ such that no $d+1$ of them are on a common hyperplane; this is equivalent to requiring that every facet of P is a d-simplex. A d-dimensional polytope P is *simple* if every point of P lies on at most than d facets; note that the points lying on exactly d facets are exactly the vertices.

The *polar* of the polytope $P = \{x \in \mathbb{R}^d \mid x^\top c_i \leq 1, \; i \in [k]\}$ with $c_i \in \mathbb{R}^d$ is $P^\Delta = \mathrm{conv}\{c_i, i \in [k]\}$. If P is simplicial or it is simple and it contains the origin $\mathbf{0}$, then $P^{\Delta\Delta} = P$; furthermore, if P is simplicial P^Δ is simple, and vice versa.

1.2 Normal Form Games and Nash Equilibria

A *game*, first defined by von Neumann in [23], is a model of strategic interaction. A *finite normal form game* is $\Gamma = (P, S, u)$ where both P and S are finite. The former is the set of *players*; $S = \times_{p \in P}$ is the set of *pure strategy profiles* and S_p is the set of *pure strategies* of player p; we will use the notation $S_{-p} = \times_{q \neq p} S_p$. The purpose of each player $p \in P$ is to maximize their *payoff function* $u^p : S \to \mathbb{R}$, where $u = \times_{p \in P} u^p$. In the following pages, by "game" we will always mean "finite normal form game." If there are only two players, we will refer to player 1 using feminine pronouns and to player 2 using masculine ones; such games are called *bimatrix games* since they can be characterized by the $m \times n$ payoff matrices A and B, where a_{ij} and b_{ij} are the payoffs of respectively player 1 and of player 2 when the former plays her ith pure strategy and the latter plays his jth pure strategy. A bimatrix game is *zero-sum* if $B = -A$, and *symmetric* if $B = A^\top$.

A *mixed strategy* of player p is a probability distribution on S_p; it can be described as a point on the $(|S_p| - 1)$-dimensional *mixed strategy simplex* $\Delta_p = \{x \in \mathbb{R}^{|S_p|} \mid x \geq \mathbf{0},\ \mathbf{1}^\top x = 1\}$. The set of *mixed strategy profiles* is the simplicial polytope $\Delta = \times_{p \in P} \Delta_p$; we extend the payoff functions to $u^p : \Delta \to \mathbb{R}$ linearly.

A *Nash equilibrium* of a game is a strategy profile in which each player cannot improve their expected payoff by unilaterally changing their strategy; such a strategy is called a *best response*. Note that applying an affine transformation to all the payoffs does not change the Nash equilibria of the game.

Proposition 1.1. *A mixed strategy $x \in \Delta_p$ is a best response against some mixed strategy profile $y \in \Delta_{-p}$ 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 ([16]). Note that there might be more than one equilibrium.

Theorem 1. (Nash [16]) *Every finite game in normal form has a Nash equilibrium.*

We give two classic examples of games: the prisoners' dilemma and a coordination game.

Example 1.1. In the symmetric non zero-sum *prisoners' dilemma* of Figure 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.

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

Figure 1.1 The prisoners' dilemma.

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.

Figure 1.2 shows a *coordination* game. Both players drive on a mountain road; they lose if drive on the same side of the road and win if they avoid each other, regardless of which side they take.

The pure strategy Nash equilibria are (mountain,valley) and (valley,mountain); there is also a symmetric equilibrium in mixed strategies at $((1/2, 1/2), (1/2, 1/2))$.

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

Figure 1.2 A coordination game.

1.3 Some Complexity Classes

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

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

where h is the *halting state*, YES is the *accepting state*, NO is the *rejecting state*, and $\{\leftarrow, \rightarrow, -\} \not\subseteq (K \cup \Sigma)$ correspond to the *cursor directions* "left," "right" and "stay."

A *language* is a set of strings of symbols $L \subseteq (\Sigma \setminus \{\sqcup\})^*$; a 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 $x \notin L$. We say that $\mathcal{M}$ *accepts* L if for every $x \in (\Sigma \setminus \{\sqcup\})^*$ we have that $\mathcal{M}(x) = \text{YES}$ if $x \in L$ and $\mathcal{M}(x)$ does not halt if $x \notin L$. Given a function $f : (\Sigma \setminus \{\sqcup\})^* \rightarrow \Sigma^*$, we say that $\mathcal{M}$ *computes* f if $\mathcal{M}(x) = f(x)$ for every $x \in (\Sigma \setminus \{\sqcup\})^*$. A *class* is a set of languages.

A specific input of a problem is called an *instance*. If a problem P outputs either YES or NO, P is a *decision problem*; its *complement* is the problem

$\overline{P}$ that outputs "NO" for each instance of P that outputs "YES", and vice versa. A *function problem* outputs a string y, more generic than YES or NO, that satisfies a binary relation $R(x, y)$, where x is the instance of P; a *search problem* either outputs y as above or it rejects the input if it's not possible to find any such y. If y is guaranteed to exist, the problem is called a *total function problem.*

An example of decision problem is: "(input) given a graph, (question) is it possible to find an Euler tour of the graph?" Its complement is "(input) given a graph, (question) is it possible that there isn't any Euler of the graph?" A search problem is: "(input) given a graph, (output) return one Euler tour of the graph, or "NO" if no such tour exists." A total function problem is: "(input) given an Euler graph, (output) return one of its Euler tours."

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*, denoted $P_1 \leq_{\mathrm{P}} P_2$, if there exists a *polynomial-time reduction*, that is, a function $f : \{0,1\}^* \to \{0,1\}^*$ and a Turing machine $\mathcal{M}$ such that for all $x \in \{0,1\}^*$

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

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 "difficult to solve"; then the problem P_1 is at least as "difficult."

For any class C of decision problems, the class of all complements of the problems in C is the *complement class* co $-$ C. A problem P is *hard* for a class C if every problem in C is polynomial-time reducible to P; that is, if P is hard to solve at least as every problem in C. A *complete* for the class C is a C$-$*hard* problem that is also in C.

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

1. for all $x \in P$ there exists a *certificate* $y \in \{0,1\}^*$ which satisfies $|y| \leq p_1(|x|)$;
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.

Intuitively, a decision problem is in **NP** if it takes polynomial time to verify whether the "certificate solution" y is, indeed, a correct answer to the question posed by the problem. The class **#P** is the class of all problems that output the number of possible certificates for a problem in **NP**.

In [13], Megiddo and Papadimitriou introduced the classes **FNP**, *function non-deterministic polynomial*, and **TFNP**, *total function non-deterministic polynomial*. The former is defined as the class of binary relations $R(x, y)$ such that there is a polynomial-time algorithm that decides $R(x, y)$ for x, y such that $|y| \leq p(|x|)$, where p is a polynomial. The latter is the class of all such problems for which y is guaranteed to exist. **FNP** and **TFNP** can be seen as the equivalent of **NP** for (respectively) function and total function problems. Also in [13], Megiddo and Papadimitriou proved that **TFNP** is a *semantic* class, that is, a class without complete problems, unless $\mathbf{NP} = \mathbf{co-NP}$. To circumvent this limitation, Papadimitriou [18] focused on the problems for which the existence of a solution is proved by a specific argument, introducing the classes **PPA** (*Proof by Parity Argument*) and **PPAD** (*Proof by Parity Argument, Directed version*).

The existence of a solution for a problem in **PPA** can be proved using the argument "in any undirected graph with one odd-degree node there must be another odd-degree node." It is interesting to note that **PPA**-complete problems are yet to be found. Problems in **PPAD**, on the other hand, are guaranteed to have a solution by a proof employing the argument "in any directed graph with one unbalanced node (that is, with outdegree different from its indegree) there must be another unbalanced node." This can be simplified to the argument "in any directed graph in which all vertices have indegree and outdegree at most one, if there is a *source* (a node with indegree zero), then there must be a *sink* (a node with outdegree zero)." Formally, we can define **PPAD** as the class of problems reducible to the problem End Of The Line. This is the definition given in Daskalakis, Goldberg and Papadimitriou [5]; the original definition in Papadimitriou [18] is given in terms of Turing machines.

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

A *circuit* is formally defined as a directed acyclic graph with n vertices with indegree 0 called *input nodes*, m vertices with outdegree 0 called *output nodes*, and *internal nodes* with indegree 1 or 2; when each input node receives an input in $\{0,1\}$, the internal nodes with indegree 2 compute the Boolean functions *and* or *or*, the internal nodes with indegree 1 compute the Boolean function *not* and each output node returns a value in $\{0,1\}$ accordingly. The problems in **PPAD** can be seen as a circuit S ("successor"), and a circuit P ("predecessor") that are used to build a directed graph with an edge (x,y) if and only if $S(x) = y$ and $P(y) = x$, with a *standard source* 0^n; the output is either a sink or a non-standard source. Figure 1.3 presents an example of graph of a **PPAD** problem; a graph for a **PPA** problem is analogous, but the resulting graph is not directed and instead of sources and sinks there are generic endpoints.

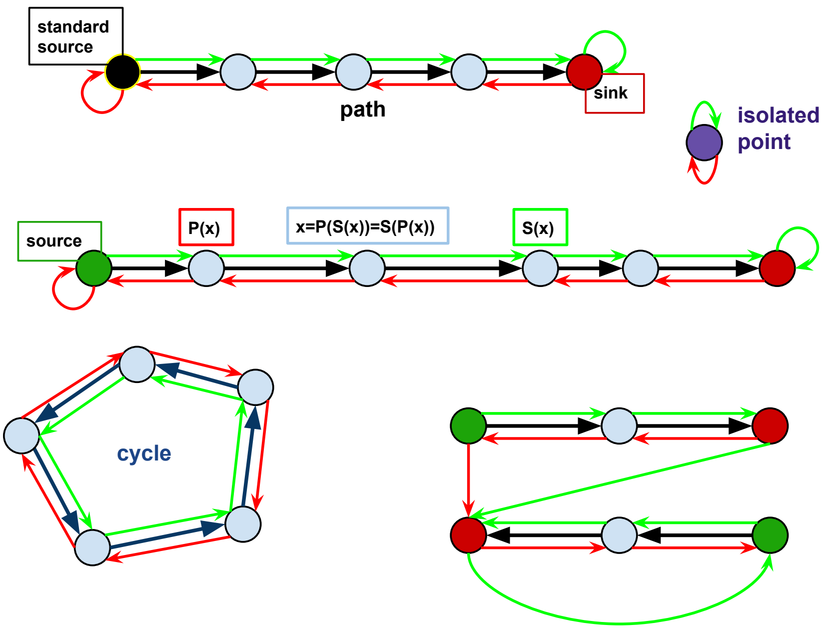

Figure 1.3 In green, the circuit S; in red, the circuit P. The standard source is the black node; the green nodes are the other sources; the red nodes are the sinks. The graph can include paths, cycles and isolated points.

By theorem 1, the problem n-Nash, defined as follows, is a total function problem.

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

Megiddo and Papadimitriou ([13]) proved that n-Nash is in **TFNP**. Daskalakis, Goldberg and Papadimitriou [5] and Chen and Deng [4] have later proven its

PPAD-completeness, the former for $n \geq 3$ and the latter for $n \geq 2$. A small amendment of the proof in [5] can be found in Casetti [2].

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

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

Chapter 2

Labels, Polytopes and Gale Strings

In the rest of this thesis we will focus bimatrix games, analizing their Nash equilibria with the use of labels and combinatorial structures. This idea is due to Lemke and Howson [11]; we will follow the approach given by Shapley [21]. We will first relate the labeling construction that allows to identify Nash equilibria with "completely labeled" vertices or facets of certain polytopes; we will then move on to unit vector games, a special class of games that allows to simplify the labeling construction; see Balthasar [1] and Savani and von Stengel [20]. In the second section of the chapter we will restrict our scope to Gale games, a further case of unit vector games. The labeling problems for these games are equivalent to an elegant combinatorial problem on a case of bitstrings satisfying some simple conditions, called Gale strings. We will define Gale strings, see their correspondence to the structures given in the first section, and finally get a reduction from 2-NASH to the Gale string problem ANOTHER GALE.

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

2.1 Bimatrix Games and Labels

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

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

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

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

1. the m pure strategies of player 1 are identified by $1, \dots, m$;
2. the n pure strategies of player 2 are identified by $m + 1, \dots, m + n$;
3. each mixed strategy $x \in X$ of player 1 has
    - label i for each $i \in [m]$ such that $x_i = 0$, that is if in x player 1 does not play her i-th pure strategy,
    - 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 $y_j = 0$, that is if in y player 2 does not play his j-th pure strategy,
    - label i for each $i \in [m]$ such that the i-th pure strategy of player 1 is a best response to y.

The labeling of mixed strategy profiles can be used to characterize the Nash equilibria of the game.

Theorem 3. (Shapley [21]) *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; this, in turn, is a necessary and sufficient condition for player 2 to play his j-th strategy at an equilibrium against x. Therefore, at an equilibrium (x, y) all labels $m + j$, with $j \in [n]$, will appear either as labels of x or of y, and analogously for the strategies $y \in Y$. □

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

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

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

The pure strategies of player 1 are labeled as $1, 2, 3$; the pure strategies of player 2 are labeled as $4, 5, 6$. In the following figures the labels of the strategies will be represented as circled numbers. Figure 2.1 shows X and Y: the exterior facets are labeled with the pure strategy on the opposite vertex, where only that pure strategy is played; the interior is covered by the best response regions, by to the other player's pure best response strategies. For example, the best-response region in Y with label 1 is the set of those (y_1, y_2, y_3) so 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 $1, 2, 6$, so this is the only Nash equilibrium of the game.

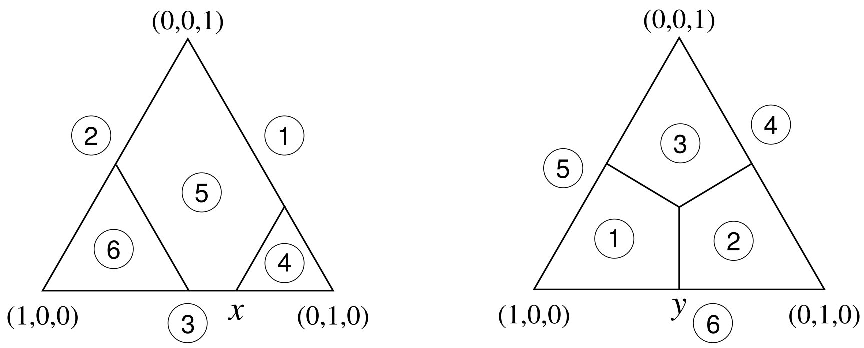

Figure 2.1 Mixed strategy sets X and Y of game (2.2) and their labeled best response regions.

The point of view 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

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

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

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

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

Figure 2.2 shows the best-response facets of $\overline{P}$ and their projection to X by ignoring the payoff variable v, which gives the subdivision of X into best-response regions of Figure 2.1.

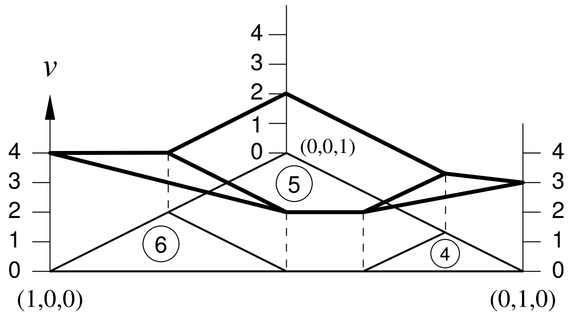

Figure 2.2 Best response polyhedron $\overline{P}$ of game (2.2).

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

$$P = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}; \quad Q = \{y \in \mathbb{R}^n \mid y \geq \mathbf{0},\ Ay \leq \mathbf{1}\}. \quad (2.4)$$

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

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

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

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

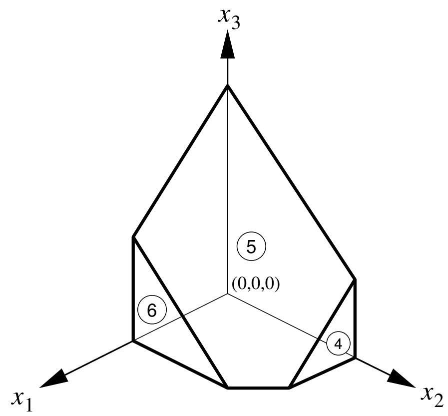

Figure 2.3 Best response polytope of game (2.2).

We will now see some special cases of games connected by polynomial-time reductions; our goal is to find a class of games that captures the complexity of 2-NASH. First of all, we note that any bimatrix game can be "symmetrized"; the result is due to Gale, Kuhn and Tucker [10] for zero-sum games and its extension to non-zero-sum games is a folklore result.

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

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

McLennan and Tourky [12] have proven a result in the opposite direction of proposition 2.1: any symmetric game can be translated into a bimatrix game of the form (I, B), called *imitiation game*. Since it takes polynomial time in the size of a matrix to calculate its transpose, we have a polynomial-time reduction from 2-NASH to IMITATION NASH.

Theorem 4. (McLennan and Tourky [12]) *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 $B = C^\top$.*

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. Note, though, that Theorem 4 applies to the symmetric equilibria of the symmetric game, but not to all its equilibria; there could be non-symmetric equilibria of $(C, C^\top)$ that are not found through the imitation game. We see an example illustrating this.

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

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

so that the corresponding imitation game is $(I, C^\top) = (A, B)$, as in (2.2). 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 corresponding imitation game (A, B) has only one equilibrium (x, y), corresponding to (x, x).

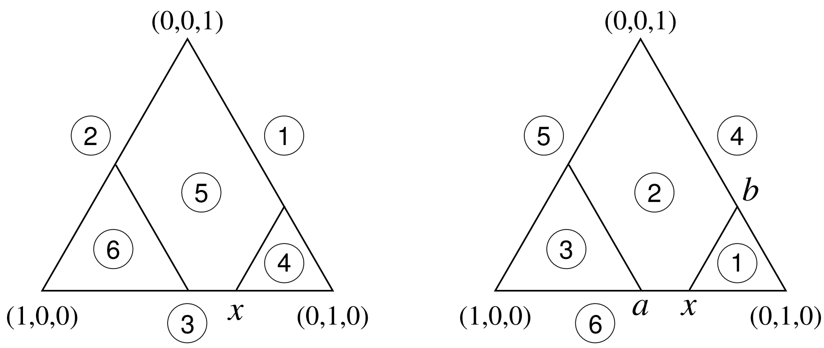

Figure 2.4 Best response regions of the symmetric game (2.7).

The characterization of Nash equilibria as completely labeled pairs (x, y) holds for arbitrary bimatrix games. From now on, we will also impose a further condition: that all points in P have at most m labels, and all points in Q have at most n labels. These games are called *nondegenerate*; since any game can be made nondegenerate by lexicographic perturbation (see von Stengel [24]), we can impose the nondegeneracy condition without loss of generality. In an equilibrium (x, y) of a nondegenerate game each label appears exactly once; this also means that the number of pure best response strategies against a mixed strategy is never larger than the size of the support of that mixed strategy. Geometrically, this means that no point of the best response polytope P lies on more than m facets and no point of the best response polytope Q lies on more than n facets, so both P and Q are simple. Furthermore, a point of P has exactly m labels if and only if it is a vertex, and a point of Q has exactly n labels if and only if it is a vertex; therefore, all completely labeled points (x, y) are vertices, and Nash equilibria are isolated points.

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

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

As it is 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. Note that 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, as shown in Figure 2.6.

A generalization of imitation games is the class of *unit vector games*, introcuced by Balthasar [1]; they 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; that is, 2-Nash reduces to Unit Vector Nash. In unit vector games, the problem of finding a completely labeled vertex of the polytope $P \times Q$ can be translated to the problem

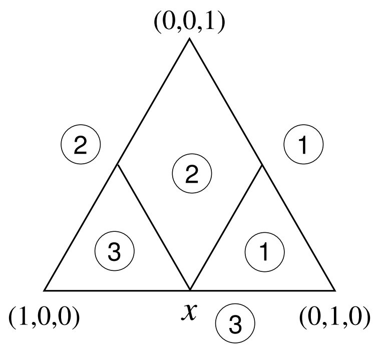

Figure 2.5 Best-response regions of the degenerate symmetric game (2.8). Label 3 appears twice at the unique symmetric equilibrium.

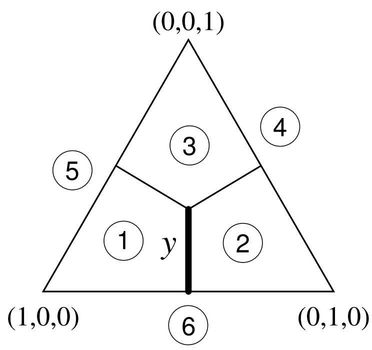

Figure 2.6 Labeled mixed-strategy sets for the imitation game $(I, C^\top)$. The equilibria (x, y) corresponding to the unique symmetric equilibrium of the symmetric game (2.8) are not unique.

of finding a completely labeled vertex of one simple polytope that encodes all the game. We first give this result as in Savani and von Stengel [20]; we will later see the version in Balthasar [1].

Theorem 5. (Savani and von Stengel [20]) *Let $l : [n] \to [m]$, and let (U, B) be the unit vector game with $U = (e_{l(1)} \ \cdots \ e_{l(n)})$. Let $N_i = \{j \in [n] \mid l(j) = i\}$ for $i \in [m]$, and consider the polytopes P^l and Q^l*

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

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

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

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

Proof. Let P, Q be the best response polytopes of (U, B) as in 2.4, and let $(x, y) \in P \times Q \setminus \{\mathbf{0}, \mathbf{0}\}$ be a Nash equilibrium of (U, B). Then (x, y) is completely labeled in $[m + n]$; so if $x_i = 0$, then x has label $i \in m$. If $x_i > 0$, then y has label i, so $(Uy)_i = 1$; then 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 $j \in [m]$ such that $(B^\top x) = j$ and $l(j) = i$; that is, $j \in N_i$. For all i such that $x_i > 0$, define y as follows: $y_j = 1$; $y_h = 0$ for all $h \in N_i \setminus \{j\}$. Then $(x, y) \in P \times Q$ is completely labeled. □

Example 2.6. The game in Example 2.1 is a unit vector game, with $l(i) = i$. In the polytope P^l of Figure 2.7 the labels 4, 5 and 6 of the best response polytope are replaced by 1, 2 and 3, since the corresponding columns of A are the unit vectors e_1, e_2, e_3. Apart from the origin $\mathbf{0}$, the only completely labeled point of P^l is x, corresponding to the unique equilibrium of game (2.2).

We will now move on the dual version of Theorem 5, as given in Balthasar [1]. We traslate the polytope P^l of (2.9) to $P = \{x - \mathbf{1} \mid x \in P^l\}$, multiplying all payoffs in B by a constant, if necessary, so that $\mathbf{0}$ is in the interior of P.

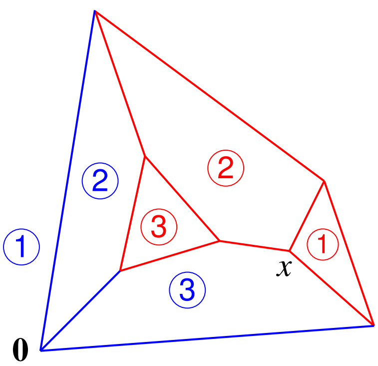

Figure 2.7 The polytope P^l of the unit vector game (2.2). Label 1 refers to the hidden facet.

We have that

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

The polar of P is then

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

where $c_j = b_j/(1 - \mathbf{1}^\top b_j)$. Since P is a simple polytope with $\mathbf{0}$ in its interior, $P^{\Delta\Delta} = P$. Furthermore, P^Δ is simplicial, therefore the facets of P^Δ 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

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

Theorem 5 then translates into the following.

Theorem 6. (Balthasar [1]) *Let Q be a labeled m-dimensional simplicial polytope with $\mathbf{0}$ in its interior and vertices $e_1, \ldots, e_m, c_1, \ldots, c_n$ such that (2.12) is a facet of Q. Let (U, B) be a unit vector game, with $U = (e_{l(1)} \ \cdots \ e_{l(n)})$ for* *a labeling $l : [n] \to [m]$ and $B = (b_1 \ \cdots \ b_n)$, where $b_j = c_j/(1 + \mathbf{1}^\top c_j)$ for $j \in [n]$. Let l_v be the labeling of the vertices of Q given by*

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

Then a facet $F \neq F_0$ of Q with normal vector v is completely labeled if and only if (x, y) is a Nash equilibrium of (U, B), where $x = (v+\mathbf{1})/(\mathbf{1}^\top(v+\mathbf{1}))$, so $x_i = 0$ 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 $j \in [n]$ such that c_j is a vertex of F.

As in Theorem 5 we have 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 $\mathbf{0}$, in Theorem 6 we have a correspondence between the completely labeled facets of the polytope Q and equilibria of (U, B) with the "artificial" equilibrium corresponding to the facet F_0 in (2.12).

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

Another Completely Labeled Vertex
input : A simple m-dimensional polytope P with $m+n$ facets; a labeling $l_f : [m+n] \to [n]$; a facet F_0 of P, completely labeled by l_f.
output: A facet $F \neq F_0$ of S, completely labeled by l.

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

Theorem 6 gives the dual of Proposition 2.2; since the construction of the polar polytope from the original one is also polynomial, the reduction is to the problem ANOTHER COMPLETELY LABELED FACET.

ANOTHER COMPLETELY LABELED FACET	
input :	A simplicial m-dimensional polytope Q with $m+n$ vertices; a labeling $l_v : [m+n] \to [n]$; a facet F_0 of Q, completely labeled by l_v.
output:	A facet $F \neq F_0$ of S completely labeled by l_v.

Proposition 2.3. 2-NASH *reduces in polynomial time to* 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 satisfies a further condition: being the dual of a cyclic polytope. Cyclic polytopes can be represented in a sintetic way using a combinatorial structure, the Gale strings. We will first give the definition of cyclic polytope, then of Gale string, then the theorem by Gale [9] about their correspondence.

The *moment curve* in dimension d is defined as

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

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

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

Example 2.7. The cyclic polytope in dimension 3 with 6 facets can be seen in figure 2.8.

Given $k \in \mathbb{N}$ and a set S, we can represent the function $f : [k] \to S$ as the string $s = s(1)s(2)\cdots s(k)$; we have a *bitstring* if $S = \{0, 1\}$. A maximal

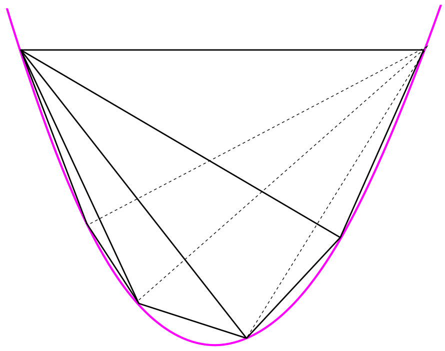

Figure 2.8 The cyclic polytope $C_3(6)$

substring of consecutive 1's in a bitstring is called a *run*; an *interior* run is bounded on both sides by 0's. We will use the notation $\mathbf{1}^k$ for a run of length k ($\mathbf{1}$'s will be in boldface for readability), and 0^k for a string of 0's of length k A *Gale string of length* n *and dimension* d, where $n > d$, is a bitstring $s \in G(d, n)$ satisfying the following conditions:

1. exactly d bits of s are equal to $\mathbf{1}$;
2. (*Gale evenness condition*)

$$0\mathbf{1}^k 0 \text{ is a substring of } s \quad \Longrightarrow \quad k \text{ is even.} \tag{2.16}$$

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

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

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

The strings $\mathbf{1111}00$, $\mathbf{111}00\mathbf{1}$, $\mathbf{11}00\mathbf{11}$, $\mathbf{1}00\mathbf{111}$, $00\mathbf{1111}$ and $0\mathbf{1111}0$ are equivalent under a cyclic shift (if considering the strings as "loops", the $\mathbf{1}$'s are all consecutive), as are the strings $\mathbf{11}0\mathbf{11}0$, $\mathbf{1}0\mathbf{11}0\mathbf{1}$ and $0\mathbf{11}0\mathbf{11}$ (if considering the strings as "loops", the even runs of $\mathbf{1}$'s are two couples separated by a single 0).

As an example of odd d, we have

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

Note that $0\mathbf{1}0\mathbf{11}$ is a cyclic shift of $\mathbf{1}0\mathbf{11}0$, but it is not a Gale string.

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

Theorem 7. (Gale [9]) *For any positive integers d, n with $n > d$*

$$\begin{aligned} &F \text{ is a facet of } C_d(n) \\ &\iff \\ &F = \operatorname{conv}\{\mu(t_j) \mid s(j) = 1 \text{ for } s \in G(d,n)\}. \end{aligned} \tag{2.17}$$

Proof. We have that $C_d(n) = \operatorname{conv}\{\mu_d(t_j) \mid t_1 < \ldots < t_n \text{ affinely independent}\}$. Let $\overline{t_1} < \ldots < \overline{t_d}$ be a choice of some of the t_j's in the definition of $C_d(n)$. Then the points $\mu_d(\overline{t_i})$, where $i \in [d]$, define an hyperplane H that crosses the moment curve $\mu_d(t)$ at all and only the $\overline{t_i}$'s. Given the definition of moment curve, the hyperplane H is never tangent to the moment curve, and every crossing gives a "change of sign"; that is, if there is one and only one $\overline{t_i} \in (t, t')$ then $\mu_d(t)$ and $\mu_d(t')$ have opposite sign. A facet F of the cyclic polytope $C_d(n)$ then corresponds to a choice of $\overline{t_i}$'s with $i \in [d]$, such that for all the

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

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

Example 2.9. Consider the facet F of the cyclic polytope $C_3(6)$ underlined in blue in Figure 2.9. If we label the vertices on the moment curve as $i \in [n]$,

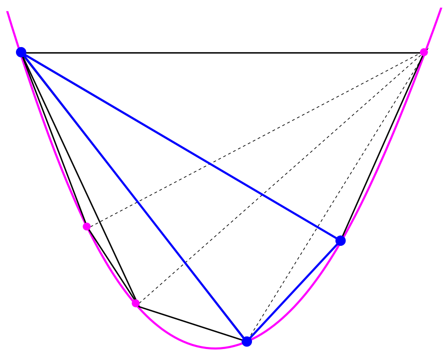

Figure 2.9 A facet of the cyclic polytope $C_3(6)$.

and we set $i = 1$ if i is a vertex of F and $i = 0$ otherwise, we see that the corresponding Gale string $s \in G(3, 6)$ is $s = \mathbf{1}00\mathbf{11}0$. On the hyperplane, the correspondence is as in figure 2.10.

Example 2.10. Figure 2.11 shows the cyclic polytope $C_4(6)$, with the exterior facet corresponding to the Gale string $s = \mathbf{1111}00$. On the hyperplane, the string $s = \mathbf{1111}00$ can be seen as in figure **??**.

Example 2.11. As a counterexample, consider figure 2.13. There are two points t_j that lie on the moment curve but are on two different sides of the hyperplane

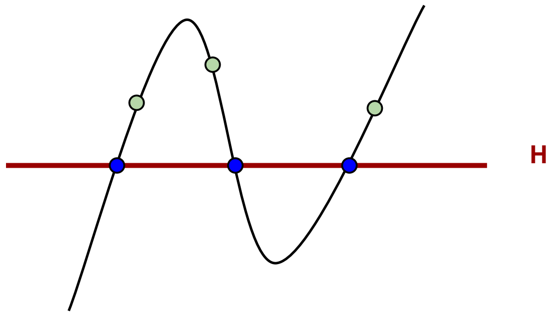

Figure 2.10 The facet of the cyclic polytope $C_3(6)$ corresponding to the Gale string $s = \mathbf{1}00\mathbf{11}0 \in G(3, 6)$ as an hyperplane intersecting the moment curve.

H, so they do not belong to the same facet. The corresponding bitstring is $s = \mathbf{111}0\mathbf{1}0$, which is not a Gale string; the violation of the Gale evenness condition correspond to the change of sign between the t_j's. An analogous case is shown in Figure 2.14; the corresponding bitstring should be $s = \mathbf{1}0\mathbf{1}0\mathbf{1}0$.

We now apply Theorem 7 to the study of bimatrix games. Proposition 2.3 states that 2-Nash can be reduced to Another Completely Labeled Facet. If the polytope Q in 6 is cyclic and we define a labeling for Gale strings such that a completely labeled Gale string corresponds to a completely labeled facet of Q, then we can study unit vector games and their dual cyclic best response polytope as Gale strings. We say that $s \in G(d, n)$ is a *completely labeled Gale string* for some labeling function $l_s : [n] \to [d]$ if $\{l(i) \mid s(i) = \mathbf{1} \text{ for } i \in [n]\} = [d]$. Since $s \in G(d, n)$ has exacty d bits equal to $\mathbf{1}$, this means that for each $j \in [d]$ there is exactly one $i \in [n]$ such that $s(i) = \mathbf{1}$ and $l_s(i) = j$. Note that it is not always possible to find a completely labeled Gale string.

Example 2.12. For $l = 121314$, there are no completely labeled Gale strings. The labels $l(i) = 2, 3, 4$ appear only once in l, as $l(2), l(4), l(6)$ respectively; therefore we must have $s(2) = s(4) = s(6) = 1$. For every other $i \in [n]$ we have $l(i) = 1$, so we have $l(i) = 1$ for exactly one $i = 1, 3, 5$. The candidate strings are then $\mathbf{11}0\mathbf{1}0\mathbf{1}$, $0\mathbf{111}0\mathbf{1}$, $0\mathbf{1}0\mathbf{111}$; but none of these satisfies the Gale

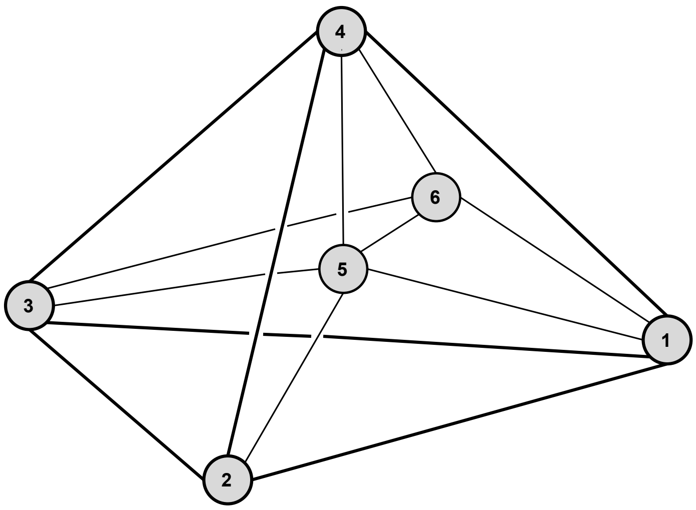

Figure 2.11 The cyclic polytope $C_4(6)$. The thin lines represent the edges inside the facet 1234, drawn in bold lines.

evenness condition.

A Gale game is a unit vector game (U, B), where $U = (e_{l(1),\ldots,e_{l(d)}})$ for some labeling $l : [n] \to [d]$, for which the dual of the best response polytope is a cyclic polytope $Q = \text{conv}\{e_1, \ldots, e_d, c_1, \ldots, c_n\}$. Theorem 6 gives a labeling l_v of the $d + n$ vertices of Q in 2.13

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

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

$$l_s(i) = i \text{ for } i \in [d];$$
$$l_s(d + j) = l(j) \text{ for } j \in [n]. \tag{2.18}$$

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

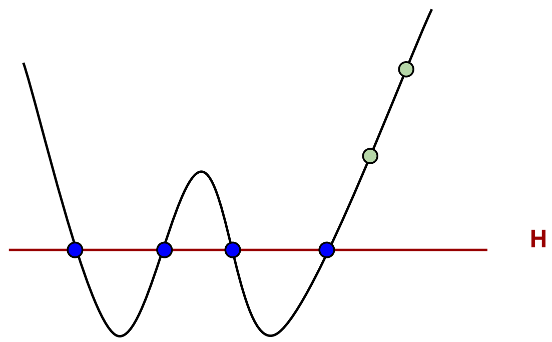

Figure 2.12 The facet of the cyclic polytope $C_4(6)$ corresponding to the Gale string $s = \mathbf{1111}0 \in G(4,6)$ as an hyperplane intersecting the moment curve.

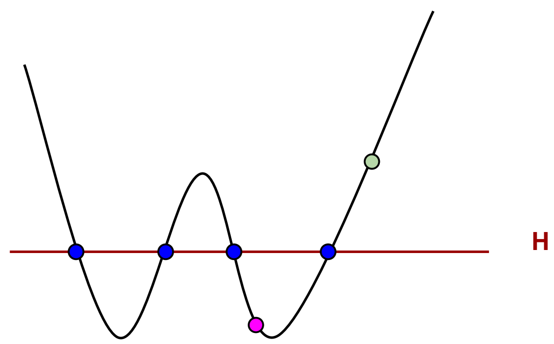

Figure 2.13 The point in pink and the point in green on the moment curve have different sign, so they don't correspond to a facet; the corresponding bitstring $s = \mathbf{111}0\mathbf{1}0$ does not satisfy the Gale evenness condition.

From this point forward, we will assume that d is even. We will also assume that the labeling $l : [n] \to [d]$ is such that $l(i) \neq l(i+1)$; this can be done without loss of generality, given the following consideration. Suppose that $l(i) = l(i+1)$ for some index i, and let s be a completely labeled Gale string for l. Then only one of $s(i)$ and $s(i+1)$ can be equal to $\mathbf{1}$ (note that it's possible that both are 0s). So $s(i)s(i+1)$ will never be a run of even length that interferes with the Gale Evenness Condition, so we can "simplify" by identifying the indices i and $i+1$.

Example 2.13. Given the string of labels $l = 123432$, there are four associ-

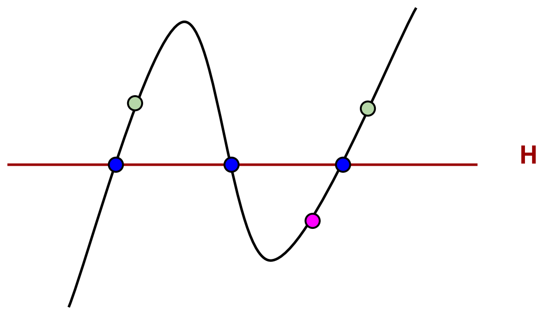

Figure 2.14 The point in pink and the point in green on the moment curve have different sign, so they don't correspond to a facet; the corresponding bitstring $s = \mathbf{1}0\mathbf{1}0\mathbf{1}0$ does not satisfy the Gale evenness condition.

ated completely labeled Gale strings in $G(4,6)$: $s_A = \mathbf{1111}00$, $s_B = \mathbf{11}0\mathbf{11}0$, $s_C = \mathbf{1}00\mathbf{111}$ and $s_D = \mathbf{1}0\mathbf{11}0\mathbf{1}$, as in Figure 2.15. On the cyclic polytope

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

Figure 2.15 The completely labeled Gale strings for the labeling $l = 123432$.

$C_4(6)$, these correspond to the facets as in figure 2.16.

We can now define the problem Another Gale as follows:

Another Gale	
input :	A labeling $l : [n] \to [$
	$s \in G(d, n)$, complete
output:	A Gale string $s' \in$
	$s' \neq s$.

It takes polynomial time to translate a cyclic polytope $C_d(n)$ into the corresponding $G(d,n)$: the bitstrings of length n with d positive bits are found in

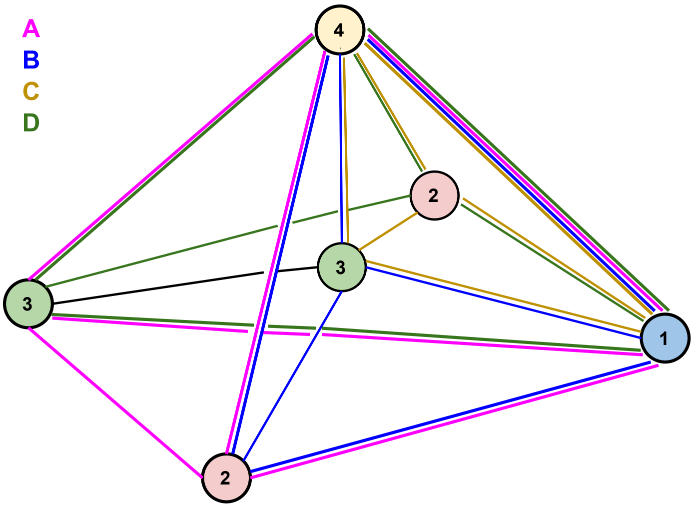

Figure 2.16 The cyclic polytope $C_4(6)$ with vertices labeled following $l_s = 123432$. The completely labeled facets correspond to the completely labeled Gale strings of Figure 2.15.

$O(n^2)$ time, and checking that a bitstring satisfies the Gale evenness condition takes $O(d)$ time, so the operation takes $O(n^3 d)$ time. Furthermore, building the labeling l_s from the labeling l_v also takes polynomial time: for the labels $i \in [m]$ it is immediate, for the labels $m + j$, where $j \in [n]$, we have to check the labeling $l : [n] \to [m]$, that is, the $n \times m$ matrix U in the imitation game. Therefore, Proposition 2.3 gives a reduction from 2-Nash to Another Gale.

Proposition 2.4. *2*-Nash *is polynomial-time reducible to* Another Gale.

Chapter 3

Algorithmic and Complexity Results

In the previous chapter we have defined some problems of the form "find another completely labeled..." for vertices, facets and Gale strings. In this section we will focus on different versions of a standard algorithm, first introduced by Lemke and Howson in [11], that yields a solution to these problems using two main concepts: the *pivoting* routine and *almost complete labeled* vertices, facets and Gale strings.

3.1 The Lemke-Howson Algorithm

In the previous chapter we have defined some problems of the form "find another completely labeled..." for vertices, facets and Gale strings. In this section we will focus on different versions of a standard algorithm, first introduced by Lemke and Howson in [11], that yields a solution to these problems using two main concepts: the *pivoting* routine and *almost complete labeled* vertices, facets and Gale strings.

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

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

Let $m, n \in \mathbb{N}$ with $m \leq n$; consider a set X with $|X| = n$ and a labeling $l : X \to [m]$. The m-uple $x = (x_1, \ldots, x_m) \in X^m$ is *almost completely labeled* if $\{j \in [n] \mid x_i = j \text{ for some } i \in [m]\} = [m] \setminus \{k\}$ for exactly one $k \in [m]$. That is, all labels appear once in x, except for the *missing label k* and a *duplicate label $h \in [m]$* that appears twice.

It's easy to see that if we pivot from an almost completely labeled facet (or vertex) on the duplicate label, or from a completely labeled facet (or vertex) on any label, we reach either an almost completely labeled or a completely labeled facet (or vertex).

We now focus on the case of pivoting on vertices of simple polytopes with labeled facets, that is, the case of the classic Lemke-Howson algorithm, first given by Lemke and Howson in [11]; we follow the very clear exposition given by Shapley in [21].

Running the Lemke-Howson algorithm defines a *Lemke path* that connects two different completely labeled vertices through almost completely labeled vertices and edges where the only missing label is k.

Algorithm 1: Lemke-Howson algorithm

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

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

1 choose any label $k \in [d]$

2 pivot on label k from x_0 to x

3 **while** *x is not completely labeled* **do**

4   pivot on the duplicate label h from x to $x' \neq x_0$

5   set $x_0 = x$, $x = x'$

6 **return** x

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

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

Proof. The fact that x is completely labeled is trivial; we must show that $x \neq x_0$. At each vertex x' of the Lemke path there are only two edges corresponding to the missing label k, since P is simple; one is the edge that has been traversed to get to x', the other one will be traversed to leave it in the next step. Therefore, there are no "loops" where a vertex is visited more than once; Lemke paths are *simple paths*.

Each Lemke path is uniquely determined by its missing label and its starting point; furthermore, the Lemke path from the endpoint with the same missing label will lead back to the starting point. Since the endpoint and the starting point are different, the Lemke paths must connect an even number of points. □

For each label $k \in [d]$ chosen in line 1 of Algorithm 1, Lemke paths can be seen as edges of an undirected graph that connect vertices corresponding to the completely labeled vertices of P, with a vertex corresponding to x_0 as standard endpoint. This shows the following proposition.

Proposition 3.2. Another Completely Labeled Vertex *is in* **PPA**.

Applying the parity result in 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 an "artificial" equilibrium, we have the following result, due to Lemke and Howson [11].

Theorem 8. (Lemke-Howson [11]) *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 way is to "symmetrize" the game as in proposition 2.1. Let $C = \left(\begin{smallmatrix} 0 & A \\ B^\top & 0 \end{smallmatrix}\right)$ and let $S = \{z \in \mathbb{R}^{m+n} \mid z \geq \mathbf{0},\ Cz \leq \mathbf{1}\}$ be the polytope associated to the game $(C, C^\top)$. We can label the $2(m+n)$ inequalities defining the facets of of S as $1, \ldots, m+n, 1, \ldots, m+n$ and apply the Lemke-Howson algorithm starting from the vertex $\mathbf{0}$; this will return a Nash equilibrium (z, z) of C, and the corresponding Nash equilibrium $(x, y) = z$ of (A, B).

We can also follow the "traditional" version of the Lemke-Howson algorithm, alternating moves on the best response polytopes P and Q as defined in 2.4, starting from the couple of vertices $(\mathbf{0}, \mathbf{0})$. Since we move in $\mathbb{R}^m$ and $\mathbb{R}^n$ instead of $\mathbb{R}^{m+n}$, this version is much easier to visualize.

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

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

The best response polytopes can be represented as the best response regions (see figure 2.1) extended to the origin $\mathbf{0}$, as in figure 3.1; the label "outside" refers to the "back" of the polytope.

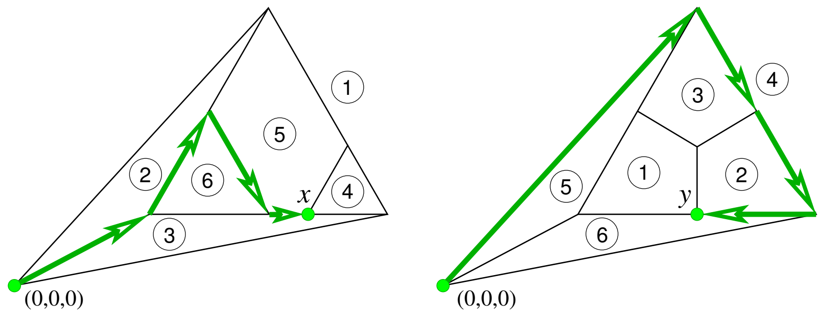

Figure 3.1 Lemke path for missing label 2 on the best response polytopes P and Q of game (2.2).

The path starts in $(\mathbf{0}, \mathbf{0})$; we drop the label 2 and we move on the polytope P. The label 6 is duplicate; so we drop the label 6 and we move on the polytope Q; an so on until we reach the point x, that is a Nash equilibrium of (A, B).

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

Algorithm 2: Dual Lemke-Howson algorithm

---

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

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

choose any label $k \in [d]$
pivot on label k from F_0 to F
while *x is not completely labeled* **do**
  pivot on the duplicate label h from F to $F' \neq x_0$
  set $F_0 = x$, $F = F'$
return x

Proposition 3.3. *The dual Lemke-Howson algorithm 2 returns a solution to the problem* Another Completely Labeled Facet.

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

We also have the analogous of proposition 3.2

Proposition 3.4. ANOTHER COMPLETELY LABELED FACET *is in* **PPA**.

Example 3.2. Consider the octahedron with vertices labeled as in figure 3.2. The facet A is completely labeled; dropping the vertex with label 1 we pivot to the completely labeled facet B.

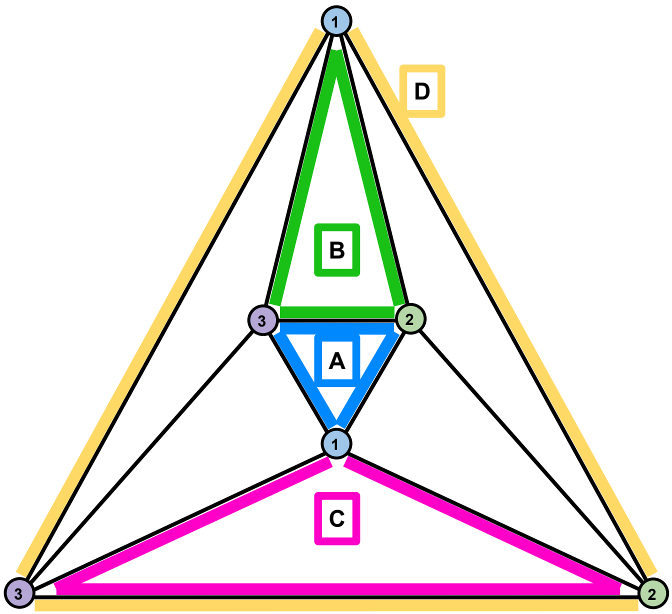

Figure 3.2 A pivot on the facets of the octahedron.

The Lemke paths for the completely labeled facets of the octahedron are shown in figure 3.3.

By theorems 5 and 6, in the case of unit vector games is enough to apply the Lemke-Howson algorithm to the polytope $P^l = \{x \in \mathbb{R}^m \mid x \geq \mathbf{0},\ B^\top x \leq \mathbf{1}\}$ in (2.9), or the dual Lemke-Howson algorithm to the polytope $Q = \text{conv}(\{e_1, \dots, e_m\}) \cup \{c_1, \dots, c_n\})$ in (2.11). Not only this yield a Nash equilibrium, but no potential solutions are "lost" considering the polytope P^l

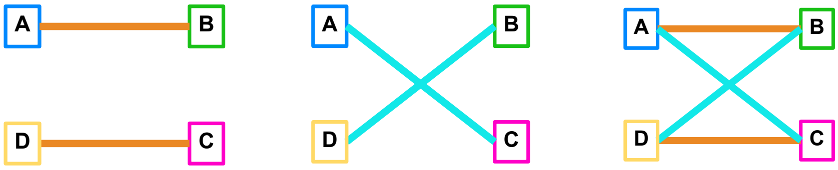

Figure 3.3 Right: Lemke paths for label 1. Centre: Lemke paths for labels 2 and 3. Left: all Lemke paths.

with m labels instead of the product of polytopes $P \times Q$ with $m+n$ labels, as stated in the following theorem by Savani and von Stengel [20]; an analogous result holds for the dual case.

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

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

We see the correspondence between the Lemke-Howson and the Lemke-Howson for Gale algorithms in the next example.

Example 3.3. Figure 3.4 shows the cyclic polytope $C_4(6)$ with the labeling given in example 2.13. This corresponds to the labeling $l = 123432$, for which

Algorithm 3: Lemke-Howson for Gale algorithm

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

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

choose a label $k \in [d]$
pivot on label k from s_0 to s
while *s is not completely labeled* **do**
    pivot on the duplicate label h from s to $s' \neq s_0$
    rename $s_0 = s$, $s = s'$
return s

there are four completely labeled Gale strings in $G(4, 6)$: $s_A = \mathbf{1111}00$, $s_B = \mathbf{11}0\mathbf{11}0$, $s_C = \mathbf{1}00\mathbf{111}$ and $s_D = \mathbf{1}0\mathbf{11}0\mathbf{1}$, corresponding to the facets A, B, C and D. Pivoting from facet A dropping label 3 yields facet B; analogously, pivoting from $s_A = \mathbf{1111}00$ dropping label 3 yields $s_B = \mathbf{11}0\mathbf{11}0$.

The analogous of Proposition 3.1 and Proposition 3.3 for the Lemke-Howson for Gale algorithm is the following.

Proposition 3.5. *The Lemke-Howson for Gale algorithm 3 returns a solution to the problem* Another Gale.

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

In the case of Gale strings, it is quite easy to orient the Lemke paths, and prove a stronger result of **PPAD** complexity instead of **PPA** as in Proposition 3.2 and Proposition 3.4. We will do so by giving a *sign*, positive or negative, to the completely labeled Gale strings; we will then show that all the endpoints of the Lemke paths have different sign, so the Lemke paths can be oriented from positive to negative sign.

A *permutation* of the elements of an ordered set S is a sequence without

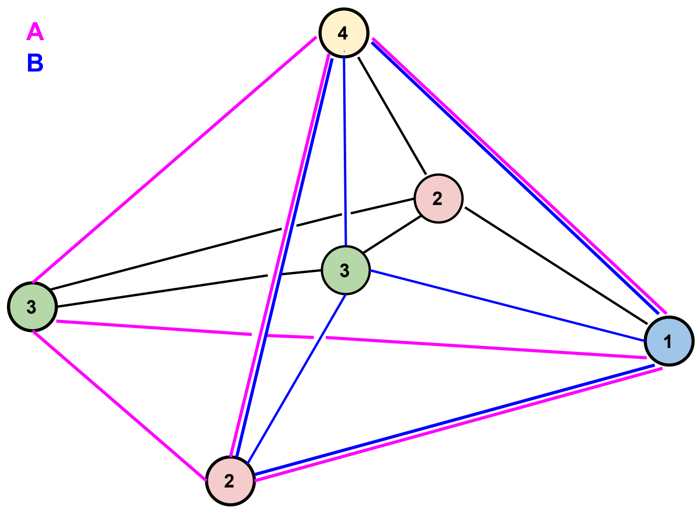

Figure 3.4 Lemke-Howson for Gale algorithm: the pivoting to s_A = **1111**00 to s_B = **110110** correspond to the pivoting from the facet A to the facet B.

repetition of elements of S. Let σ be a permutation of the elements of $[d]$. The *sign* of σ can be defined as $\mathrm{sign}(\sigma) = (-1)^m$, where m is the number of the exchanges of exactly two elements of σ (called *transpositions*) needed to get $\sigma' = 1 \ldots n$ from σ. Note that any two permutations that differ in one transposition have opposite sign.

We give the *sign of a completely labeled Gale string* s for a labeling l as follows. Let l_0 be the string of labels $l(i)$ such that $s(i) = 1$ and that two labels corresponding to a run in l are adjacent in l_0. We define $\mathrm{sign}(s) = \mathrm{sign}(l_0)$. Note that, in the context of games, the completely labeled Gale string $1^d 0^{(n-d)}$ corresponding to the artificial equilibrium will always have positive sign.

Example 3.4. For the labeling $l = 123434$ there are four completely labeled Gale string in $G(4, 6)$. The completely labeled string 111100 has positive sign, since it corresponds to the string of labels 1234; the string 110110 corresponds to the string of labels 1243, that has negative sign. The string 111001 corresponds to 4123, with negative sign, since $s(6)$ and $s(1)$ are both 1 and therefore form a run.

Defining l_0 as above, but for an almost completely labeled Gale string, we have that l_0 has one missing and one duplicate label. We can substitute the former to the latter either in the position that has been reached by the last pivot, obtaining the string l_1, or in the position that was already in the string, obtaining the string l_2. These two strings have opposite sign, since they can be obtained from each other with one transposition.

We can now give a sign to the pivoting steps of the Lemke-Howson for Gale algorithm. Note that a pivoting operation changes the sign, since it involves "jumping" over an odd number of **1**'s. Assume that the completely labeled Gale string s has positive sign (the negative case is the same with opposite signs). If the pivoting returns another completely labeled Gale string s', this must have negative sign because it has been obtained via one pivoting step. If the pivoting returns an almost completely labeled Gale string, the sign of l_1 will be negative, since it correspond to a pivoting; so the sign of l_2 will be positive. The next pivoting step drops the label that was substituted with the missing one in l_2, so again we change sign. The steps of the Lemke paths can therefore be seen as in Figure 3.5

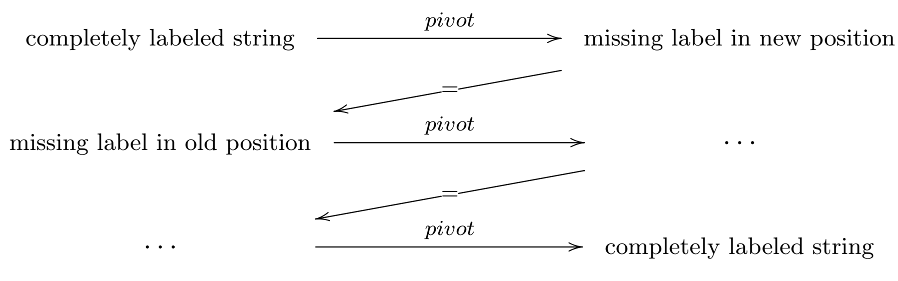

Figure 3.5 Sign switching on the Lemke-Howson for Gale algorithm.

Example 3.5. Let $l = 123432$; consider the Lemke path from the completely labeled Gale string $s = \mathbf{1111}00$ dropping label 1. The graph in Figure 3.5 then becomes as in figure 3.6

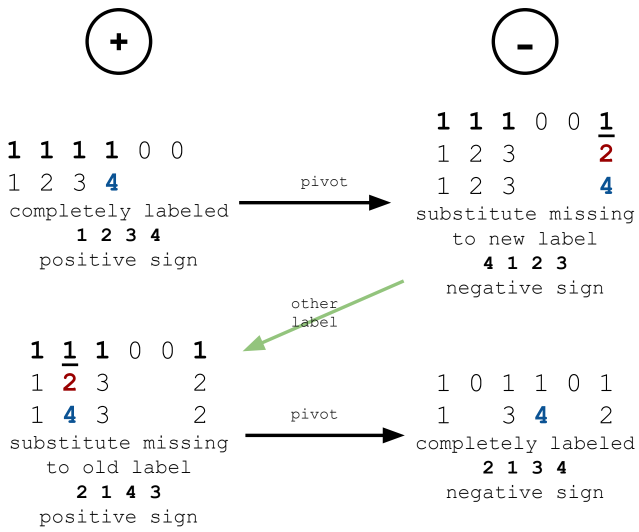

Figure 3.6 Pivoting with sign on 123432.

Proposition 3.6. Another Gale *is in* **PPAD**.

A result similar to Proposition 3.6, but more general, is given in Shapley [21]; it shows that two equilibria at the ends of a Lemke path have opposite *index*. The index is defined in terms of the signs of the determinants of the square submatries of the payoff matrices for the equilibrium support; the artificial equilibrium is assigned index $+1$. The main result of Shapley's article [21] is that if a nondegenerate game has n Nash equilibria with index $+1$, then the game has $n+1$ Nash equilibria with index -1. The article also gives an interesting example of a game for which the graph of all Lemke paths is disjoint. This is the symmetric game $(C, C^\top)$ with

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

The equilibria are $(x_1, y_1) = ((0,0,1),(0,0,1))$, $(x_2, y_2) = ((1/3,2/3,0),(1/3,2/3,0))$ and $(x_3, y_3) = ((1/6,1/3,1/2),(1/6,1/3,1/2))$. All Lemke paths from the artificial equilibrium $(0,0)$ end at (x_1, y_1). Note that since all equilibria are symmetric, by Theorem 4 they all correspond to equilibria in the imitation game $(I, C^\top)$.

An interesting example of the Lemke-Howson for Gale algorithm is the following, due to Morris [15].

Example 3.6. Consider the labeling $l = 1234564523$ for $G(6,10)$ and the completely labeled Gale string $s = \mathbf{111111}00$. Dropping the label 1, the Lemke-Howson for Gale algorithm will run as in figure 3.7.

$$
\begin{array}{cccccccccc}
\mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{4} & \mathbf{5} & \mathbf{2} & \mathbf{3} \\
\hline
\underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdot & \cdot & \cdot & \cdot \\
\cdot & \mathbf{1} & \mathbf{1} & \underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \overline{\mathbf{1}} & \cdot & \cdot & \cdot \\
\cdot & \mathbf{1} & \mathbf{1} & \cdot & \underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \overline{\mathbf{1}} & \cdot & \cdot \\
\cdot & \underline{\mathbf{1}} & \mathbf{1} & \cdot & \cdot & \mathbf{1} & \mathbf{1} & \mathbf{1} & \overline{\mathbf{1}} & \cdot \\
\cdot & \cdot & \mathbf{1} & \overline{\mathbf{1}} & \cdot & \mathbf{1} & \underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \cdot \\
\cdot & \cdot & \mathbf{1} & \mathbf{1} & \overline{\mathbf{1}} & \mathbf{1} & \cdot & \underline{\mathbf{1}} & \mathbf{1} & \cdot \\
\cdot & \cdot & \underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \cdot & \cdot & \mathbf{1} & \overline{\mathbf{1}} \\
\cdot & \cdot & \cdot & \underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \overline{\mathbf{1}} & \cdot & \mathbf{1} & \mathbf{1} \\
\cdot & \cdot & \cdot & \cdot & \underline{\mathbf{1}} & \mathbf{1} & \mathbf{1} & \overline{\mathbf{1}} & \mathbf{1} & \mathbf{1} \\
\overline{\mathbf{1}} & \cdot & \cdot & \cdot & \cdot & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
\hline
\mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} & \mathbf{6} & \mathbf{4} & \mathbf{5} & \mathbf{2} & \mathbf{3}
\end{array}
$$

Figure 3.7 Morris path on $C(6,10)$

Morris paths are exponentially long

Savani and von Stengel (2006) construct bimatrix games where both players have dual cyclic polytopes as their best response polytopes. They call these games m âĞě n-double cyclic polytope games. For square games, where m = n = d and where the labels are derived from MorrisâĂŹs construction (see Morris (1994) and Section 4.4 for the introduction of these labels), the LH-algorithm takes exponentially many steps to find an equilibrium.

However, square games are not aÌĆAÌĘIJhard to solveaÌĆAÌĘIÌĞ by the sup- port enumeration algorithm; see Savani (2006). The support enumeration algorithm considers strategies of the players with equal support size and checks whether they are best replies to each other. Since square games have a unique completely mixed equilibrium, where both players play d pure strategies with positive probability, the support enumeration algorithm terminates quickly. Games where both solution concepts take exponentially long are then called hard- to-solve bimatrix games One class of hard-to-solve games is constructed from 3d aÌĆGÌĘeÌŇ d games with one cyclic polytope of dimension d and one simplotope, which is a product of simplices, here d tetrahedra (Sa- vani (2006)). He calls these triple imitation games, due to the connection to imitation games. Nash equilibria of triple imitation games are fully de- scribed by completely labeled Gale strings in G(d,4d). They are thus Gale games and an equilibrium can be found via the problem ANOTHER COM- PLETELY LABELED GALE STRING.

This gives a strong motivation to study the complexity of Another Gale, since it seems that it relates to games that are hard to solve. Our main result, in the next section, will give a **FP** algorithm to solve it.

3.2 The Complexity of Gale and Another Gale

We will now give our main result: Another Gale can be solved in polynomial time; therefore, it takes polynomial time to find a Nash Equilibrium of a bimatrix game with dual cyclic best response polytope. Our proof will rely on the construction of a graph and, if possible, a perfect matching for it. A *perfect matching* of a multigraph $G = (V, E)$ is a set $M \subseteq E$ of pairwise non-adjacent edges so that every vertex $v \in V$ is incident to exactly one edge in M. A theorem by Edmonds ([6]) gives the complexity of the associated problem Perfect Matching.

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.

Theorem 10. (Edmonds [6]) *The problem* Perfect Matching *can be solved in polynomial time.*

To prove our main result on Another Gale, we will first focus on the accessory problem Gale, and we will use theorem **??** to prove that it is solvable in polynomial time. We will consider every Gale string as a "loop."

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

Theorem 11. *The problem* Gale *is solvable in polynomial time.*

Proof. We give a reduction of Gale to Perfect Matching.

Consider the multigraph $G = (V, E)$ with $V = [d]$, so that the vertices of G correspond to the labels $l(i) \in [d]$, and $E = \{(l(i), l(i + 1))$ for $i \in [n]\}$, so that there is an edge between two vertices if and only if the corresponding labels are next to each other at some index i. Let $s \in G(d, n)$ be a completely labeled Gale string. By Gale evenness condition, every run of s corresponds uniquely to $d/2$ pairs of indices $(i, i+1)$ with $s(i) = s(i+1) = 1$, and since s is completely labeled, all labels $l(i) \in [d]$ occur at exactly one of these indices. Then the edges $(l(i), l(i+1))$ form a perfect matching of G.

Conversely, let $l : [n] \to [d]$ be a labeling, and let M be a perfect matching for G. Consider a bitstring s 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, considering s as a "loop," every run of s is of even length, thus satisfying the Gale evenness condition. Since M is perfect, every vertex $v \in [d]$ is the endpoint of an edge $(l(i), l(i+1))$, so s has exactly d bits equal to $\mathbf{1}$, so it is completely labeled.

We have therefore reduced the problem GALEto PERFECT MATCHING, that by theorem 7 can be solved in polynomial time. □

We give two examples of the construction used in theorem 7.

Example 3.7. Figure **??** shows the graph for the Morris labeling $l = 1234564523$, and its two matchings $M = \{e_1, e_3, e_5\}$ and $M' = \{e_8, e_6, e_{10}\}$.

These, in turn, correspond to the completely labeled Gale strings $s =$ $\mathbf{111111}0000$ and $s = 1000\mathbf{11111}$.

A perfect matching for a graph, and therefore a Gale string for a labeling, is not always possible, as shown in the next example.

Example 3.8. Consider the labeling $l = 121314$. The graph G is shown in figure **??**

Since there aren't any disjoint edges, it's not possible to find a perfect matching for G. We have already seen in example 2.12 that there isn't any possible completely labeled Gale string for $l = 121314$.

We finally extend the proof of theorem 7 to ANOTHER GALE.

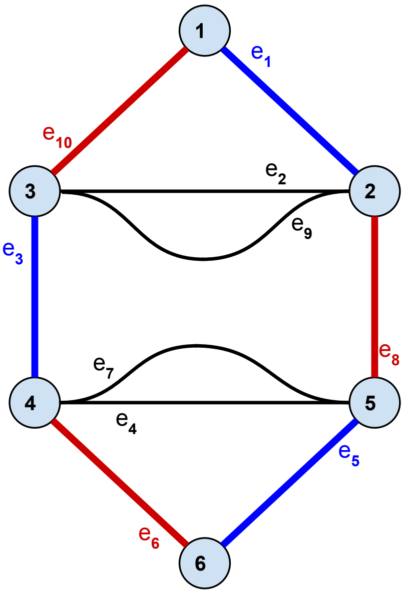

Figure 3.8 The graph G and its matchings for the Morris labeling $l = 1234564523$.

Theorem 12. *The problem* ANOTHER GALE *is solvable in polynomial time.*

Proof. Let $G = (V, E)$ be the graph corresponding to the labeling $l : [n] \to [d]$ as in the proof of theorem 7 and let M be the perfect matching of G corresponding to the completely labeled Gale string $s \in G(d, n)$.

If there are two edges $e, e' \in E$ such that $e \in M$, both e and e' have endpoints $l(i), l(i+1)$, but $e \neq e'$ (recall that G can be a multigraph), the matching $M' = (M \setminus \{e\}) \cup \{e'\}$ is perfect. The corresponding completely labeled Gale string $s' \in G(d, n)$ satisfies $s' \neq s$, since in s the $\mathbf{1}$'s corresponding to the labels $l(i), l(i+1)$ are in the positions given by the edge e, while in s' they are in the positions given by $e' \neq e$. It takes time $d/2$ to check all edges of M, the time required is still polynomial.

We now assume that all the edges in every perfect matching M for G don't have a parallel edge. Since by theorem 3.5 there is an even number of completely labeled Gale strings, the existence of s guarantees the existence of another completely labeled Gale string $s' \neq s$ and the corresponding perfect

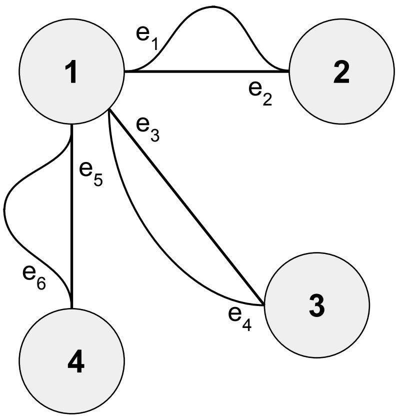

Figure 3.9 The graph for the labeling $l = 121314$

matching $M' \neq M$. Since $M' \neq M$, there is at least one edge $e' \in M$ such that $e' \notin M'$. Consider the $d/2$ graphs $G_i = (V, E_i)$, where $E_i = E \setminus \{e_i\}$ for $e_i \in M$. Since $V(G) = V(G_i)$ and $E(G_i) \subset E(G)$, every perfect matching for one of these G_i is a perfect matching for G as well. With a brute force approach, we look for a perfect matching in each G_i; this will be M'. Since there are $i \in [d/2]$, the time to find it will be still polynomial. □

We give two examples of the construction of theorem **??**.

Example 3.9. The labeling $l = 123432$ gives the graph G in figure **??**. Suppose that Edmonds' algorithm returns the matching $M = \{e_1, e_3\}$, associated to the completely labeled Gale string $s = \mathbf{1111}00$. The edge e_1 has a parallel edge, e_6; we immediately have a second perfect matching in $M' = \{e_3, e_6\}$, associated to the Gale string $s' = \mathbf{101101}$.

A case without parallel edges in the matching is the Morris graph.

Example 3.10. Consider the Morris graph of example **??**; suppose that Edmonds' algorithm returns the perfect matching $M = \{e_1, e_3, e_5\}$, as in figure **??** right, corresponding to the completely labeled Gale string $s = \mathbf{111111}0000$. We can then delete the edge e_1 to obtain the graph G_1, as in figure **??** left.

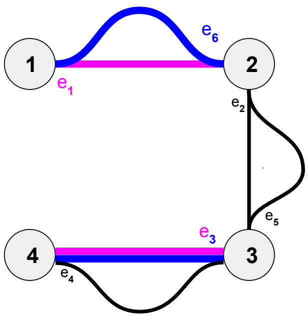

Figure 3.10 The graph for the labeling $l = 123432$.

The graph G_1 has a perfect matching $M' = \{e_6, e_8, e_{10}\}$; this is also a perfect matching of G, corresponding to the string $s' = 1000\mathbf{11111}$.

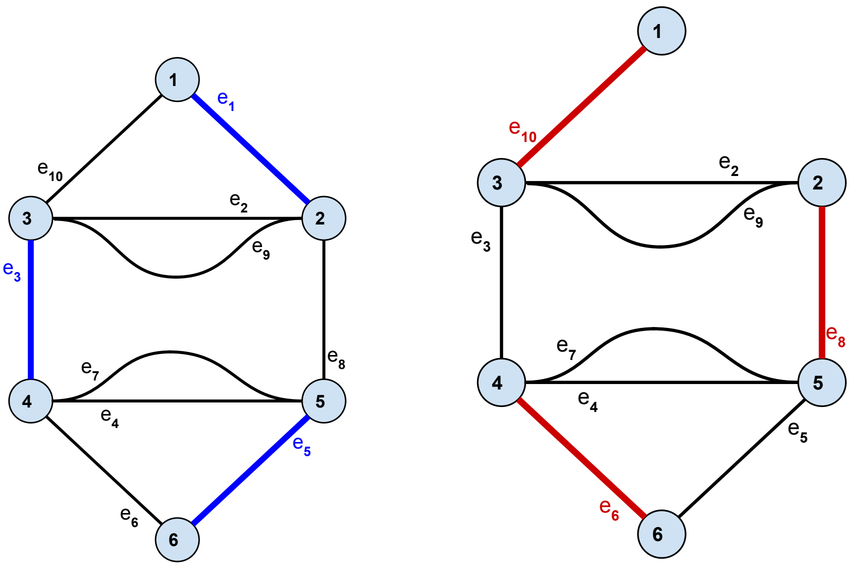

Figure 3.11 Left: the Morris graph $G = (V, E)$ with the matching $M = \{e_1, e_3, e_5\}$. Right: the graph $G_1 = (V, E \setminus \{e_1\})$ with the matching $M' = \{e_6, e_8, e_{10}\}$.

THEOREM

Finding a Nash equilibrium in Gale games takes polynomial time.

Proof: Consider the labeled cyclic polytope of the Gale game as described in Section 3.2. The artificial Nash equilibrium is given by a completely labeled facet F0 and has a corre- sponding completely labeled Gale string in the Gale representation of the polytope. Since finding ANOTHER COMPLETELY LABELED GALE STRING is in FP, another completely labeled facet F1, which corresponds to a Nash equlibrium, is found in polynomial time. Since all reductions are polynomial in the size of the problem, a Nash equilibrium in Gale games is found in polynomial time. This result has an interesting implication. The Nash equilibrium that is found via the translation to perfect matchings above is not necessarily the same Nash equilibrium that is found by the LHG-algorithm for Gale strings. This holds because EdmondsâĂŹs algorithm picks out one of the other perfect matchings while the LHG-algorithm finds a specific one, described in Section 3.4.

Chapter 4

Further results

P complexity of finding all (?)

from SvS-15

Ve ÌĄgh and von Stengel (2014, Thm. 12) give a near-linear time algorithm that finds such a second perfect matching that, in addition, has opposite sign, which corresponds to a Nash equilibrium of positive index as it would be found by a Lemke path (which, however, can be exponentially long). So this combinatorial problem is simpler than the problem of finding a Nash equilibrium of a bimatrix game, even though it gives rise to games that are hard to solve by the standard methods considered in Theorem 11.

from VvS

This paper presents three main contributions in this context. First, we define an abstract framework called pivoting systems that describes âĂIJcomplementary pivoting with directionâĂİ in a canonical manner. Similar abstract pivoting systems have been proposed by Todd (1976) and Lemke and Grotzinger (1976); we compare these with our approach in Section 5. Second, using this framework, we extend the concept of orientation to oiks and show that room partitions at the two ends of a pivoting path have opposite sign, provided the underlying oik is oriented. For two-dimensional oiks, which are Euler graphs, room partitions are perfect matchings. Their orientation is the sign of a perfect matching as defined for Pfaffian orientations of graphs. Our third result is a polynomial-time algorithm for the following problem: Given a graph 2 with an Eulerian orientation and a perfect matching, find another perfect matching of opposite sign. The complementary pivoting algorithm that achieves this may take exponential time.

We conclude with open questions on the computational complexity of pivoting sys- tems. Consider a labeled oriented pivoting system whose components (in particular the pivoting operation) are specified as polynomial-time computable functions. Assume one CL state is given. The problem of finding a second CL state belongs to the com- plexity class PPAD (Papadimitriou, 1994). This problem is also PPAD-complete, because finding a Nash equilibrium of a bimatrix game is PPAD-complete (Chen and Deng, 2006), which is a special case of an oriented pivoting system by Proposition 1. However, there should be a much simpler proof of this fact because pivoting sys- tems are already rather general, so that it should be possible to encode an instance of the PPAD-complete problem âĂIJEnd of the LineâĂİ (see Daskalakis, Goldberg, and Papadimitriou, 2009) directly into a pivoting system. Finding a Nash equilibrium of a bimatrix game is PPAD-complete, and LemkeâĂŞ Howson paths may be exponentially long. Savani and von Stengel (2006) showed this with games defined by dual cyclic polytopes for the payoff matrices of both players, and a simpler way to do this is to use the Lemke paths by Morris (1994). One motivation for the study of Casetti, Merschen, and von Stengel (2010) was the question if finding a second completely labeled Gale string is PPAD-complete. This is unlikely because this problem can be solved in polynomial time with a matching algorithm. For the complexity class PPADS, where one looks for a second CL state of opposite sign (Daskalakis, Goldberg, and Papadimitriou, 2009), this problem is also solvable in polynomial time with our algorithm of Theorem 12. However, for room partitions of 3-oiks, already manifolds, finding a second room partition is likely to be more complicated. Is this problem

PPAD-complete? We leave these questions for further research.

Acknowledgements

appendix/acknowledgments

Bibliography

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

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

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

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

[5] C. Daskalakis, P. W. Goldberg, C. H. Papadimitriou (2006). "The Complexity of Computing a Nash Equilibrium." *SIAM Journal on Computing*, 39(1), pp. 195–259.

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

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

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

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

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

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

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

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

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

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

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

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

[18] 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.

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

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

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

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

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

[24] 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.

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

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