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 [25]

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 [24] . 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 [25] 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 $\text{conv}\{z_1, \ldots, z_m\} = \{\sum_i \lambda_i z_i \mid \lambda_i \geq 0,\ \sum_i \lambda_i = 1\}$. A set of points $Z = \{z_1, \ldots, z_m\}$ is *convex* if $Z = \text{conv}(Z)$, 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 $\Delta_d = \text{conv}\{e_1, \ldots, e_{d+1}\}$.

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 $P = \text{conv}(Z)$ is a point $z \in Z$ such that $\text{conv}(Z \setminus \{z\}) \neq P$; an *edge* of P is a 1-dimensional line segment that has two vertices as endpoints. A *facet* of P is the convex hull of a set of d vertices $F = \{z_1, \ldots, z_d\}$ 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 $P^\Delta = \text{conv}\{c_i, i \in [k]\}$. 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	0 3
help	3 0	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 $((1/2, 1/2), (1/2, 1/2))$.

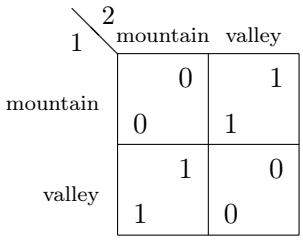

Figure 1.2 A coordination game.

1.3 Some Complexity Classes

A *Turing machine* $\mathcal{M}$ is a representation of a *program* 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*); δ is the *transition function*

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

where h is the *halting state*, YES is the *accepting state*, *No* is the *rejecting state*, and $\{\leftarrow, \rightarrow, -\} \nsubseteq (K \cup \Sigma)$ correspond to the *cursor directions* "left," "right" and "stay."

A *language* is a set of strings of symbols $L \subset (\Sigma \setminus \{\sqcup\})^*$; a Turing machine $\mathcal{M}$ *decides* L if for every $x \in (\Sigma \setminus \{\sqcup\})^*$ $\mathcal{M}(x) = Yes$ if $x \in L$ and $\mathcal{M}(x) = No$ if $x \notin L$. We say that $\mathcal{M}$ *accepts* L if for every $x \in (\Sigma \setminus \{\sqcup\})^*$ $\mathcal{M}(x) = Yes$ if $x \in L$ and $\mathcal{M}(x)$ does not halt if $x \notin L$. Given a function $f : (\Sigma \setminus \{\sqcup\})^* \to \Sigma^*$, we say that $\mathcal{M}$ *computes* f if for every $x \in (\Sigma \setminus \{\sqcup\})^*$ $\mathcal{M}(x) = f(x)$.

Given a problem, a specific input of a problem is called an *instance*. The output of a *decision problem* P is either YES or NO. Its *complement* is the problem $\bar{P}$ that outputs "NO" for each instance of P that outputs "YES", and

vice versa. A *function problem* outputs a more generic y that satisfies a binary relation $R(x, y)$, where x is the instance of the problem. *Search problems* are function problems that return either y such that $R(x, y)$, or "NO" 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 $P_1 \leq_P P_2$, 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."

A *class* is a set of languages. For any class C of decision problems, the class of all complements of the problems in C is the *complement class* $\text{co} - \text{C}$. A problem P is *hard* for a class C if for every problem P_C in C there is a polynomial-time reduction to P; that is, if P is hard to solve at least as every problem in C. A $\text{C} - \textit{hard}$ problem in C is *complete* for 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. Intuitively, **FNP** and **TFNP** are similar to **NP**, but they allow for problems of (respectively) function and total function form. Also in [13], Megiddo and Papadimitriou proved that, unless $\mathbf{NP} = \mathbf{co} - \mathbf{NP}$, **TFNP** is a *semantic* class, that is, a class without complete problems. To circumvent this limitation of **TFNP**, Papadimitriou

[18] focused on the problems for which the existence of a solution is proved by a "parity argument", introducing the classes **PPA** (*Proof by Parity Argument*) and **PPAD** (*Proof by Parity Argument, Directed version*).

Intuitively, **PPA** is the class of problems for which the existence of a solution 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 no **PPA** problems have been proven **PPA**-complete. The existence of the problems in **PPAD**, on the other hand, can be proven using the argument "in any directed graph with one unbalanced node (that is, with outdegree different from its indegree) there must be another unbalanced node." The latter can simplified without loss of generality or computational power to the case of indegree and outdegree at most one, that is, "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.

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 $S(P(x)) \neq x \neq 0^n$

This is the definition given in Daskalakis, Goldberg and Papadimitriou [5]; Papadimitriou [18] defines the class in terms of Turing machines. 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. Problems in **PPA** can be defined analogously, such that the resulting graph is not directed and instead of sources and sinks there are generic endpoints. We have that PPAD $\subset$ PPA, and there are **PPA** problems that are not **PPAD**, as we will see in section **??**. Figure 1.3 presents an example of graph of a **PPAD** problem; a graph for a **PPA** problem is analogous, but undirected.

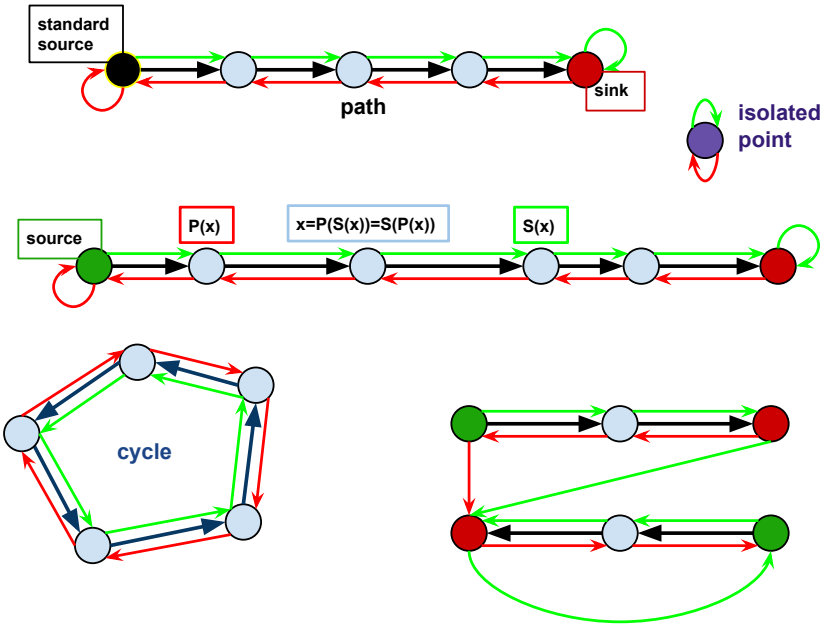

Figure 1.3 In green, the circuit S; in red, the circuit P. The standard source is the black node; the green nodes are the other sources; the red nodes are the sinks. The graph can include paths, cycles and isolated points.

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 it is in **TFNP**. Daskalakis, Goldberg and Papadimitriou [5] and Chen and Deng [4] have 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 **??**; in fact, our main result can be seen as a negative result on the **PPAD** complexity of a case of 2-Nash.

Chapter 2

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 (2012). "Completely Labeled Facet is NP-complete." Manuscript, 6 pp.

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