Abstract

This thesis presents a report on original research, published as joint work with Merschen and von Stengel in *Electronic Notes in Discrete Mathematics* (2010). Our result shows a polynomial time algorithm to find a Nash equilibrium for a particular class of games, which was previously used by Savani and von Stengel (2006) as an example of exponential time for the classical Lemke-Howson algorithm for bimatrix games (1964).

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 (1994) was, in turn, to capture the pivoting technique of many results related to the Nash equilibrium, including the Lemke-Howson algorithm. A PPAD-completeness proof of the games we consider would have provided a traceable proof of the Daskalakis, Goldberg and Papaditrimiou (2005) and Chen and Deng (2009) results about the PPAD-completeness of every normal form game. Our result of polynomial-time solvability, on the other hand, indicates the existence of 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 a string of labels and an associated string of 0s and 1s satisfying some conditions, called *Gale conditions*, as seen in Savani and von Stengel (2006). The second one is the polynomial-time solvability of the problem of finding a perfect matching in a graph, solved by Edmonds (1965).

Further results by Merschen (2012) and Végh and von Stengel (2014) solved the open problem of the *sign* of the equilibrium found in polynomial time.

TODO

1 Introduction

2 Bimatrix Games and Polytopes

A (d-dimensional) *simplicial polytope* P is the convex hull of a set of at least $d+1$ points v in $\mathbb{R}^d$ in general position, that is, no $d+1$ of them are on a common hyperplane.

If a point v cannot be omitted from these points without changing P then v is called a *vertex* of P. A *facet* of P is the convex hull $\operatorname{conv} F$ of a set F of d vertices of P that lie on a hyperplane $\{x \in \mathbb{R}^d \mid a^T x = a_0\}$ so that $a^T u < a_0$ for all other vertices u of P; the vector a (unique up to a scalar multiple) is called the *normal vector* of the facet. We often identify the facet with its set of vertices F.

The following theorem, due to Balthasar and von Stengel [**?**, **?**], establishes a connection between general labeled polytopes and equilibria of certain $d \times n$ bimatrix games (U,B).

THIS FROM VvS

Theorem 1 *Consider a labeled d-dimensional simplicial polytope Q with $\mathbf{0}$ in its interior, with vertices $-e_1,\ldots,-e_d,c_1,\ldots,c_n$, so that $F_0 = \operatorname{conv}\{-e_1,\ldots,-e_d\}$ is a facet of Q. Let $-e_i$ have label i for $i \in [d]$, and let c_j have label $l(j) \in [d]$ for $j \in [n]$. Let (U,B) be the $d \times n$ bimatrix game with $U = [e_{l(1)} \cdots e_{l(n)}]$ and $B = [b_1 \cdots b_n]$, where $b_j = c_j/(1+\mathbf{1}^\top c_j)$ for $j \in [n]$. Then the completely labeled facets F of Q, with the exception of F_0, are in one-to-one correspondence to the Nash equilibria (x,y) of the game (U,B) as follows: if v is the normal vector of F, then $x = (v+\mathbf{1})/\mathbf{1}^\top(v+\mathbf{1})$, and $x_i = 0$ if and only if $-e_i \in F$ for $i \in [d]$; any other label j of F, so that c_j is a vertex of F, represents a pure best reply to x. The mixed strategy y is the uniform distribution on the set of pure best replies to x.*

In the preceding theorem, any simplicial polytope can take the role of Q as long as it has one completely labeled facet F_0. Then an affine transformation, which does not change the incidences of the facets of Q, can be used to map F_0 to the negative unit vectors $-e_1,\ldots,-e_d$ as described, with Q if necessary expanded in the direction $\mathbf{1}$ so that $\mathbf{0}$ is in its interior.

A $d \times n$ bimatrix game (U,B) is a *unit vector game* if all columns of U are unit vectors. For such a game B with $B = [b_1 \cdots b_n]$, the columns b_j for $j \in [n]$ can be obtained from c_j as in Theorem 1 if $b_j > \mathbf{0}$ and $\mathbf{1}^T b_j < 1$. This is always possible via a positive-affine

transformation of the payoffs in B, which does not change the game. The unit vectors $e_{l(j)}$ that constitute the columns of U define the labels of the vertices c_j. The corresponding polytope with these vertices is simplicial if the game (U,B) is nondegenerate [**?**], which here means that no mixed strategy x of the row player has more than $|\{i \in [d] \mid x_i > 0\}|$ pure best replies. Any game can be made nondegenerate by a suitable "lexicographic" perturbation of B, which can be implemented symbolically.

Unit vector games encode arbitrary bimatrix games: An $m \times n$ bimatrix game (A,B) with (w.l.o.g.) positive payoff matrices A,B can be symmetrized so that its Nash equilibria are in one-to-correspondence to the symmetric equilibria of the $(m+n) \times (m+n)$ symmetric game (C^T,C) where

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

In turn, as shown by McLennan and Tourky [**?**], the symmetric equilibria (x,x) of any symmetric game (C^T,C) are in one-to-one correspondence to the Nash equilibria (x,y) of the "imitation game" (I,C) where I is the identity matrix; the mixed strategy y of the second player is simply the uniform distribution on the set $\{i \mid x_i > 0\}$. Clearly, I is a matrix of unit vectors, so (I,C) is a special unit vector game.

TODO - UNTIL HERE

3 Cyclic Polytopes and the Problem ANOTHER GALE

3.1 Cyclic Polytopes and Gale Strings

A special case of games is obtained by taking a particular case of best response polytope in theorem 1.

Definition 1 *A* cyclic polytope *P in dimension d with n vertices is the convex hull of distinct points $\mu(t_j)$, where $j \in [n]$ and μ is the* moment curve

$$\mu : t \mapsto (t, t^2, \ldots, t^d)^\top$$

Cyclic polytopes can be represented through a combinatorial structure, the *Gale strings*. This makes their study particularly interesting, and, as we will see, it can be used to obtain very elegant proofs.

Definition 2 *For any integer k and any set S, we can represent the function $f_s : [k] \to S$ as the string $s = s(1)s(2)\cdots s(k)$. If $S = \{0,1\}$ we denote*

$$\begin{aligned} 1(s) &= s^{-1}(1) \\ &= \{j \in [k] \mid s(j) = 1\} \end{aligned}$$

The indicator function of $1(s)$ will then correspond to a bitstring *s, a sequence of 0's and 1's.*

A maximal substring of consecutive 1's in a bitstring is called a run.

Example Let $k = 6$, and let $f_s(j) = 0$ if j is even and $f_s(j) = 1$ if j is odd. Then $s = 101010$ and $1(s) = 1,3,5$.

We can now give the definition of *Gale string*.

Definition 3 *We denote as $G(d,n)$ the set of all bitstrings s of length n such that*

1. *exactly d bits in s are 1 and*
2. *s fulfills the* Gale evenness condition*:*

$$01^k0 \text{ is a substring of } s \quad \Rightarrow \quad k \text{ is even.}$$

An element of of $G(d,n)$ is called a Gale string of dimension d and length n.

Definition **??** characterises Gale strings as bitstrings of length n with exactly d elements equal to 1, such that *interior* runs (that is, runs bounded on both sides by 0s) must be of even length. Note that this condition allows Gale strings to start or end with an odd-length run.

This leads to an important consequence when d is even.

Property 3.1 *Let d be even, and let s in $G(d,n)$. Then if s starts with an odd run it will also end with an odd run, and if s starts with an even run it will end with an even run.*

That is, the set of Gale strings of even dimension is therefore invariant under a cyclic shift of the strings.

We can then consider the Gale strings in $G(d,n)$ with even d as a "loop" obtained by "glueing together" the extremes of the string to form an even run.

Example We consider $G(4,6)$. We have

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

The strings 111100, 111001, 110011, 100111, 001111 and 011110 are equivalent under a cyclic shift, as are the strings 110110, 101101 and 011011.

The relation between cyclic polytopes and Gale strings is given by the following theorem by Gale **??**.

Theorem 2 **??** *For any positive integer n, assume that $t_1 < t_2 < \cdots < t_n$ and let P be the cyclic polytope obtained by taking t_j, where $j \in [n]$, in definition 1.*

Then the facets of P are encoded by $G(d,n)$; that is, F is a facet of P if and only if

$$F = \text{conv}\{\mu(t_i) \mid i \in 1(s)\} \qquad \text{for some } s \in G(d,n)$$

From this point forward, we will assume that d is even.

3.2 Labeling and the Problem ANOTHER GALE

Definition 4 *Given a set G of bitstrings of length n and a parameter d, a* labeling *is a function $l : [n] \to [d]$. A string s in $G(d,n)$ is* completely labeled *if $l(1(s)) = [d]$. Any $l(i) \in [d]$ is called a* label

If s is a completely labeled Gale string s associated with the labeling $l : [n] \to [d]$, then for each label $l(i)$ there is a bit $s(i) = 1$; therefore, we have at least d bits such that $s(i) = 1$, that is $|l(1(s))| = d$. Furthermore, there are exactly d bits such that $s(i) = 1$ if and only if for each label $l(i)$ there is exactly one bit $s(i) = 1$.

Example Given the string of labels $l = 123432$, there are four associated completely labeled Gale strings: 111100, 110110, 100111 and 101101.

123432	**12343**2	1**23432**	**123432**
111100	**1101**10	1**0011**1	**1011**01

Example For $l = 121314$, there are no completely labeled Gale strings.

TODO - FROM HERE

For this cyclic polytope P, a labeling $l : [n] \to [d]$ can be understood as a label $l(j)$ for each vertex $\mu(t_j)$ for $j \in [n]$. A completely labeled Gale string s therefore represents a facet F of P that is completely labeled.

Special games are obtained by using cyclic polytopes in Theorem 1, suitably affinely transformed with a completely labeled facet F_0. When Q is a cyclic polytope in dimension d with $d+n$ vertices, then the string of labels $l(1)\cdots l(n)$ in Theorem 1 defines a labeling $l' : [d+n] \to [d]$ where $l'(i) = i$ for $i \in [d]$ and $l'(d+j) = l(j)$ for $j \in [n]$. In other words, the string of labels $l(1)\cdots l(n)$ is just prefixed with the string $1\,2\cdots d$ to give l'. Then l' has a trivial completely labeled Gale string 1^d0^n which defines the facet F_0. Then the problem ANOTHER GALE defines exactly the problem of finding a Nash equilibrium of the unit vector game (I,B). Note again that B is here not a general matrix (which would define a general game) but obtained from the last n of $d+n$ vertices of a cyclic polytope in dimension d.

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

TODO - UNTIL HERE

4 Main Complexity Results

4.1 Pivoting, Parity and the Lemke-Howson Algorithm

Definition 5 *Let $l : [d] \to [n]$ be a labeling. An* almost completely labeled *Gale string associated with l is $s \in G(d,n)$ such that $|l(1(s))| = d-1$*

If s is an almost completely labeled Gale string s associated with the labeling $l : [n] \to [d]$, then for each label $l(i)$ but one there is a bit $s(i) = 1$. Furthermore, there will be exactly one "duplicate" label $l(i)$ such that $s(i) = s(j) = 1$ for exactly one $j \neq i$

Completely and almost completely labeled Gale strings are used to build two fundamental algorithms: the *pivoting algorithm* and the *Lemke-Howson for Gale algorithm*, that uses the pivoting as a subroutine.

Definition 6 *We define as* pivoting *the operation defined in algorithm 1, where we consider the Gale strings as "loops" by identifying position i with any position $i+kn$*

Algorithm 1: Pivoting

input : A string of labels l of length n; a Gale string s that describes a a complete or almost complete labeling of l; $i \in [n]$ such that $s(i) = 1$

output: A different Gale string s' that describes a complete or almost complete labeling of l.

set $s' = s$
set $s'(i) = 0$ (we call this *dropping the label $l(i)$*)
let j be the length of the odd maximal run of 1s created by this
if *the odd maximal run of* 1*s is on the* ***right*** *of position i* **then**
  set $s'(i+j+1) = 1$
else
  set $s'(i-j-1) = 1$
return s'

Example Let $l = 123432$, and let's consider the associated completely labeled Gale string $s = 111100$. Then dropping the label 1 we pivot to the almost completely labeled Gale string 011110.

1$\underline{\mathbf{1}}$**23**$\underline{\mathbf{4}}$32	drop $\underline{\mathbf{1}}$ from l
1$\underline{\mathbf{1}}$**11**$\underline{\mathbf{0}}$0	take the corresponding $\underline{1}$ in s
$\underline{0}$**11**$\underline{\mathbf{1}}$0	set it to $\underline{0}$
$\underline{0}$**1111**$\underline{\mathbf{0}}$	and set the other end of the odd run in s to $\underline{1}$
1**234**$\underline{\mathbf{3}}$2	the labels **2**, **3** (twice) and **4** in l correspond to a 1 in s

By dropping the label 2 instead, we pivot to the completely labeled Gale string 101110.

1$\underline{\mathbf{2}}$**34**32	drop $\underline{\mathbf{2}}$ from l
1$\underline{\mathbf{1}}$**11**00	take the corresponding $\underline{1}$ in s
1$\underline{\mathbf{0}}$**11**00	set it to $\underline{0}$
1$\underline{\mathbf{0}}$**11**$\underline{\mathbf{0}}$**1**	and set the other end of the odd run (on the loop) in s to $\underline{1}$
1**234**3$\underline{\mathbf{2}}$	**all the labels** in l correspond to a 1 in s

Note that, by the Gale evenness condition, we must have an the odd maximal run of 1s either on the right or on the left of position i.

If the Gale string in the pivoting algorithm is completely labeled, we can drop any label $l(i)$ such that $s(i) = 1$. We refer to these labels as *free labels*.

Each pivoting can be seen as a step of a "path" between (almost) completely labeled Gale strings. If the first and last step of the path are completely labeled Gale strings, the path is described by the Lemke-Howson for Gale algorithm.

Example Let's consider the label string $l = 123432$ and the associated completely labeled Gale string $s = 111100$, as in example 4.1. The Lemke-Howson for Gale algorithm using 1 as free label returns the completely labeled Gale string $s' = 110110$.

Algorithm 2: Lemke-Howson for Gale Algorithm

input : A n-string $l \in [n]$ where d is even and $d < n$; a completely labeled Gale string s associated with l.

output: A Gale string $s' \neq s$ associated with l.

set $s' = s$

pivot any free label of s'

while *s' is an almost completely labeled Gale string* **do**

  pivot the duplicate label, not picked up by the previous pivot

return s'

	12**3**432	drop **1**
	111**1**00	pivot
	0**1**11**1**0	
pick **3**, duplicate	1**2**3**4**3**2**	drop the other **3**
	0**1**1**1**10	pivot
	11**0**1**1**0	
pick **1**, starting label	**1**2**3**43**2**	end

Using algorithm 2 we can show a fundamental property of Gale strings.

Theorem 3 *For any labeling $l : [n] \to [d]$, where d is even and $d < n$, the number of completely labeled Gale strings associated with l is even.*

Proof. If there are no completely labeled Gale strings associated with l, the theorem holds trivially.

Suppose now that there is at least one completely labeled Gale strings associated with l.

First of all, note that a pivot is reversible. Suppose that we pivot on the (almost) completely labeled Gale string s by dropping the label $l(i)$ and picking up the label $l(j)$. Then $s(j) = 0$ and it is adjacent to the opposite side of the odd maximal run of 1s starting at i that was created by dropping $l(i)$. Let s' be the (almost) completely labeled Gale string obtained from this pivot. Analogously, if we pivot on s' by

dropping $l(j)$, we will have to pick up the label $l(i)$. The pivoting is therefore reversible by simply dropping the label that was picked up.

As there are only a finite number of possible bitstrings for each label string, if cycling is not possible the algorithm must terminate by finding another completely labeled Gale string in a finite number of steps.

TODO: edit rest of proof

Cycling is not possible due to the following observations. Suppose the algorithm returns to a bit assignment of s other than the initial Gale string. Then at this bit assignment of s, because each pivot is reversible, we would have to be able to pick up two labels. This, however, is ruled out by the GEC as only one of the adjacent runs of the dropped label is odd. Returning to initial position is only possible by reversing the initial pivot which is not allowed. The only free choice we have is at the beginning of the algorithm where we drop one free label. From then on the process of the algorithm is uniquely determined, thus terminating in a finite number of steps at another Gale string.

□

The reversibility of the pivoting steps leads to the following property

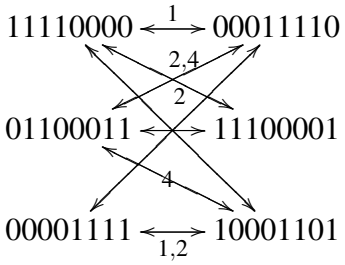
Property 4.1 *Let $s \in G(d,n)$ be a completely labeled Gale string for a labeling l, and let s' be the completely labeled Gale string obtained by running the Lemke-Howson algorithm on s by dropping the label $l(i)$. Then running the Lemke-Howson algorithm on s' by dropping the label $l(i)$ returns s.*

The converse does not hold: it's possible for two Gale strings s and s' to be the endpoints of the path given by the Lemke-Howson algorithm run by dropping both the label $l(i)$ and $l(j) \neq l(i)$. This is trivially true since it's possible that $|\{s \in G(d,n) completely labeled for l\}| < d$.

An interesting example is the following.

Example For the labeling $l = 12342314$, the completely labeled Gale strings in $G(4,8)$ are 11110000, 00011110, 00001111, 11100001, 01100011, 10001101. They are related as endpoints of the Lemke-Howson algorithm as shown in the following graph:

TODO: graph more readable



Note that the graph is bipartite: this holds in general, as it will be shown in section **??**.

Note also that it's not a complete bipartite graph: to reach 1110001 from 0000111 we must apply the algorithm three times.

To our knowledge it is an open question if the graph has to be connected.

Note that the parity result of theorem 3 is about *completely labeled* Gale strings, not Gale strings $s \in G(d,n)$ in general. For example, $|G(4,6)| = 9$, as shown in 3.1.

Theorem 3 holds for odd d as well.

4.2 The Complexity of GALE and ANOTHER GALE

We consider the following decision problem.

GALE
input: A labeling $l : [n] \to [d]$, where d is even and $d < n$.
question: Is there a completely labeled Gale string s in $G(d,n)$ associated with l?

Theorem 3 implies that if there is one completely labeled Gale string, there is also a second one. The following function problem asks to compute a completely labeled Gale string if one such string is already given.

We now show that both GALE and ANOTHER GALE can be solved in polynomial time. The proof uses a reduction to the following problem, which was first

shown to be solvable in polynomial time by Edmonds [**?**].

PERFECT MATCHING
input: A graph $G = (V, E)$.
question: Is there 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?

Theorem 4 *The problems* COMPLETELY LABELED GALE STRING *and* ALMOST COMPLETELY LABELED GALE STRING *can be solved in polynomial time.*

Proof. We give a rather simple reduction to PERFECT MATCHING. Given the labeling $l : [n] \to [d]$, construct the (multi-)graph G with vertex set $V = [d]$ and up to n (possibly parallel) edges with endpoints $l(i), l(i+1)$ for $i \in [n]$ whenever these endpoints are distinct (so G has no loops); here we let $n+1 = 1$ ("modulo n") so that $n, n+1$ is to be understood as $n, 1$. Then a completely labeled Gale string s in $G(d,n)$ splits into a number of runs which are uniquely split into $d/2$ pairs $i, i+1$ so that the labels $l(i)$ and $l(i+1)$ are distinct, and all labels $1, \ldots, n$ occur among them. So this defines a perfect matching for G.

Conversely, a perfect matching M of G defines a Gale string s where $s(i) = s(i+1) = 1$ if the edge that joins $l(i)$ and $l(i+1)$ is in M and $s(i) = 0$ otherwise, so s is completely labeled. This shows how COMPLETELY LABELED GALE STRING reduces to PERFECT MATCHING. Finding a perfect matching, or deciding that G has none, can be done in polynomial time [**?**].

The reduction for ANOTHER GALE is an extension of this. Consider the given completely labeled Gale string s and the matching M for it. If G has multiple edges between two nodes and one of them is in M, simply replace that edge by a parallel edge to obtain another completely labeled Gale string s'. Hence, we can assume that M has no edges that have a parallel edge. Another completely labeled Gale string s' exists by Theorem **??**. The corresponding matching M' does not use at least one edge in M. Hence, at least one of the $d/2$ graphs G which have one of

the edges of M removed has a perfect matching M', which is a perfect matching of G, and which defines a completely labeled Gale string s' different from s. The search for M' takes again polynomial time. □

The significance of Theorem **??** is to be understood in the context of equilibrium computation for games, which we discuss next. The remainder of this paper contains only known results.

5 Further results

Appendix A: Notation

For a matrix A we denote its transpose with A^T. We treat vectors u, v in $\mathbb{R}^d$ as column vectors, so $u^T v$ is their scalar product. By $\mathbf{0}$ we denote a vector of all 0's, of suitable dimension, by $\mathbf{1}$ a vector of all 1's. A unit vector, which has a 1 in its ith component and 0 otherwise, is denoted by e_i. Inequalities like $u \geq \mathbf{0}$ hold for all components. For a set of points S we denote its convex hull by $\operatorname{conv} S$.

For $n \in \mathbb{N}$ we denote $[n] = 1, 2, \ldots, n$

Appendix B: A result about PPAD completeness of NASH

References