Abstract

not dates but cite to bibliography

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 solve two problems related to labeled Gale strings, a combinatorial structure consisting a string of labels and a bitstring satisfying certain conditions.

Gale strings can be used in the representation of a particular class of games that Savani and von Stengel (2006) used as an example of exponential running time for the classical Lemke-Howson algorithm to find a Nash equilibrium of a bimatrix game (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.

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 (2005) and Chen and Deng (2009) 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 a string of labels and a bitstring, 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) will be mentioned.

An appendix relates an amendment to the proof of the PPAD-completeness result by Daskalakis, Goldberg and Papaditrimiou (2005).

1

1 Introduction

2 Games, Polytopes and Gale Strings

2.1 Games and Nash Equilibria

2.2 Bimatrix Games and Best Response Polytopes

possibly one subsection only

2.3 Cyclic Polytopes and Gale Strings

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

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*

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

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

$$\mathtt{1}(s) = s^{-1}(1)$$
$$= \{j \in [k] \mid s(j) = 1\}$$

The indicator function of $\mathtt{1}(s)$ 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 2.1. 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*:

$$01^k 0 \text{ is a substring of } s \quad \Rightarrow \quad k \text{ is even.}$$

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

Definition 3 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 2.1. *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.*

use "modulo" - even more than "cyclic shift"

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

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

change mention of "cyclic shift" if not used before

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

here to end subsect: polytopes

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

Theorem 1 ([**?**]). *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

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

sketch of pf if not too long and it uses relevant techniques

graphics of cyclic polytope - cfr vS articles and talks

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

give something to generalise to odd case

4

2.4 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 $l(\mathbb{1}(s)) = [d]$. Any $l(i) \in [d]$ is called a *label*

If $s \in G(d,n)$ is completely labeled for the labeling $l : [n] \to [d]$, then for each label $l(i)$ there is a bit $s(i) = 1$. We therefore have exactly d positions i for which $s(i) = 1$; hence, $|l(\mathbb{1}(s))| = d$.

Example 2.3. Given the string of labels $l = 123432$, there are four associated completely labeled Gale strings: 111100, 110110, 100111 and 101101.

123432	**123432**	123432	123432
111100	**110110**	**100111**	**101101**

Sometimes there aren't any completely labeled Gale strings that are associated with a given labeling.

Example 2.4. For $l = 121314$, there are no completely labeled Gale strings.

here to end subsect: polytopes

graphics of labeled cyclic polytope

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 **??**, 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 **??** 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 $1^d 0^n$ 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 $l : [n] \rightarrow [d]$, 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 $s' \neq s$.

---

3 Algorithmic and Complexity Results

3.1 Pivoting

3.2 The Lemke-Howson Algorithm and Parity

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 the *Lemke-Howson for Gale algorithm*. We first define its fundamental subroutine, the *pivoting algorithm*.

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 $i + kn$

use "modulo" notation

for clgs, is dropped label necessarily double one?

The operation where we set $s(i) = 0$ (line 1) is called *dropping label i*.

Example 3.1. Let $l = 123432$. The Lemke-Howson for Gale algorithm applied to the completely labeled Gale string $s = 111100$ and the label $\underline{\underline{4}}$ in position 4 returns the almost completely labeled Gale string 011110.

123<u>4</u>32	drop the label **4** from l
111<u>1</u>00	taking the corresponding **1** in s
111<u>0</u>00	and setting it to **0**
111001	set the other end of the odd run in s to **1**
1<u>2</u>343<u>2</u>	there is a **1** in s in a position corresponding to the labels **1**, **2** (twice) and **3** in l; there are only **0**'s in all the positions corresponding to the label **4** in l

---
Algorithm 1: Pivoting on completely labeled Gale strings
---

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

output: A complete or almost complete Gale string for l.

1 set $s(i) = 0$

2 let j be the length of the odd maximal run of 1s created by this

3 if *the odd maximal run of* 1*s is on the* **right** *of position* i **then**

4   set $s(i + j + 1) = 1$

5 else

6   set $s(i - j - 1) = 1$

7 return s

Example 3.2. Let $l = 123424$. The Lemke-Howson for Gale algorithm applied to the almost completely labeled Gale string $s = 110011$ and the duplicate label $\underline{\underline{2}}$ in position 2 returns the almost completely labeled Gale string 100111.

1$\underline{2}$3424	drop the label **2** in position 2 from l
110011	taking the corresponding **1** in s
1$\underline{0}$0011	and setting it to **0**
100111	set the other end of the odd run (on the loop) in s to **1**
123$\underline{\underline{4}}2\underline{4}$	there is a **1** in s in a position corresponding to the labels **1**,
	2 and **4** (twice) in l; there are only **0**'s in all the positions
	corresponding to the label **3** in l

If we drop duplicate label $\underline{\underline{2}}$ in position 5 instead, the algorithm returns the completely labeled Gale string 111001.

123$\underline{2}$4	drop the label **2** in position 5 from l
1100$\underline{1}$1	taking the corresponding **1** in s
1100$\underline{0}$1	and setting it to **0**
111001	set the other end of the odd run (on the loop) in s to **1**
12342$\underline{4}$	all the labels in l correspond to a **1** in s

8

Note that, by the Gale evenness condition, we must have an the odd maximal run of 1's 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" through (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*.

Definition 7. [?]

We define the *Lemke-Howson for Gale* algorithm as follows:

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.

1 set $s' = s$

2 pivot any free label of s'

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

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

5 **return** s'

Example 3.3. Let's consider the label string $l = 123432$ and the associated completely labeled Gale string $s = 111100$, as in example **??**. The Lemke-Howson for Gale algorithm using 1 as free label returns the completely labeled Gale string $s' = 110110$.

start	123432	drop 1
	111100	pivot
	011110	
pick 3, duplicate	123432	drop the other 3
	011110	pivot
	110110	
pick 1, starting label	123432	end

Note how the last label that is picked up is the one dropped at the start, that's been missing in all the intermediate step; otherwise, we would have reached a completely labeled Gale string at an earlier iteration.

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

Theorem 2. *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.

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.

$\square$

Theorem 2 holds for odd d as well.

expand - or cut?

The reversibility of the pivoting steps leads to the following property

Property 3.1. *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 3.4. 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:
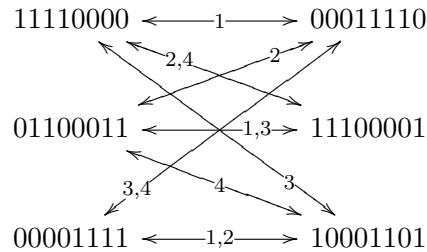
Note that the graph is bipartite: this holds in general, a property related to further results that we will discuss in section 4.

Note also that it's not a complete bipartite graph: there isn't an edge between 1110001 and 00001111.

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

If this were the case, there would be Nash equilibria not reachable via LHG from artificial equilibrium

Note that the parity result of theorem 2 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 2.2.

results on $11234 = 1234$ and $12345236 = 123423$ (can delete successive double - any length of run, can delete single occurrences if in odd run). The first one plays a part in main thms

3.3 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 for which the best response polytope is cyclic.

Our proof will be based on a simple graph construction, and it will exploit the following result by Edmonds **??** on the problem PERFECT MATCHING, defined as follows.

PERFECT MATCHING

---

input : A graph $G = (V, E)$.

output: Whether there 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.

Theorem 3 (??). *The problem* PERFECT MATCHING *is solvable in polynomial time.*

We begin by considering the accessory problem GALE, and proving that it is solvable in polynomial time.

GALE

---

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

output: Whether there is a completely labeled Gale string s in $G(d, n)$
associated with l.

Theorem 4. *The problem* GALE *is solvable in polynomial time.*

Proof. proof from old draft: edit!

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 [?].

$\square$

Two examples of the construction used in theorem 5 follows. In the first one, there is a perfect matching and a corresponding Gale string.
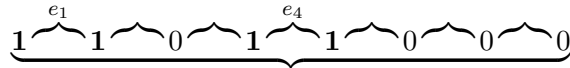
Example 3.5. Let $l = 12343122$ be a string of labels. Then we have the edges e_i as follows:
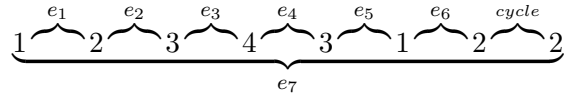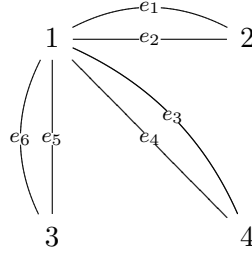

So the graph G will be:
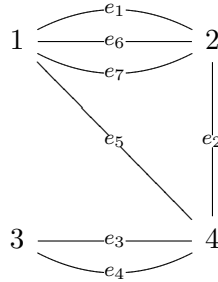
A possible completely labeled Gale string for l is 11011000, which corresponds to e_1, e_4 in G:

An example where there isn't a perfect matching, and therefore there isn't any possible Gale string for the labeling is the following.

Example 3.6. Let us consider the labeling $l = 121314$. The associated graph G will be

It's trivial to see that it's not possible to find a perfect matching for G.

We finally extend the proof of theorem 5 to show that ANOTHER GALE is polynomial-time solvable.

Theorem 5. *The problem* ANOTHER GALE *is solvable in polynomial time.*

Proof. proof from old draft: to edit!

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. $\square$

examples for second PM (one w/ double edges, one without)

15

4 Further results

Appendix A: A result about PPAD completeness of Nash

Appendix B: Notation

For a matrix A we denote its transpose with A^T.

Vectors u, v in $\mathbb{R}^d$ as column vectors

$u^T v$ is their scalar product.

$\mathbf{0}$ 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 $\text{conv}\, S$.

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

References