1 Introduction

2 Complexity, Games, Polytopes and Gale Strings

2.1 The Complexity Classes P and PPAD

2.2 Normal Form Games and Nash Equilibria

file: background-subsection

2.3 Bimatrix Games and Best Response Polytopes

file: polytopes-subsection

2.4 Cyclic Polytopes and Gale Strings

2.5 Labeling and the Problem Another Gale

file: gale-def-subsection

3 Algorithmic and Complexity Results

3.1 Pivoting and the Lemke-Howson Algorithm

label: almost-completely-labeled 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*.

label: pivoting 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?

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.

set $s(i) = 0$
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

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$\underline{\underline{4}}$32	drop the label **4** from l
111$\underline{1}$00	taking the corresponding **1** in s
111$\underline{0}$00	and setting it to **0**
111001	set the other end of the odd run in s to **1**
$\underline{\underline{1}}2343\underline{\underline{2}}$	there is a **1** in s in a position corresponding to the labels **1**, **2** $\underline{\text{(twice)}}$ and **3** in l; there are only **0**'s in all the positions corresponding to the label **4** in l

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.

$\underline{\underline{1}}2342\underline{\underline{4}}$	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**
$\underline{1}$2342$\underline{\underline{4}}$	there is a **1** in s in a position corresponding to the labels **1**, **2** and **4** $\underline{\text{(twice)}}$ 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.

$\underline{\underline{1}}2342\underline{\underline{4}}$	drop the label **2** in position 5 from l
1100$\underline{\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**
$\underline{1}$2342$\underline{4}$	$\underline{\text{all the labels}}$ in l correspond to a **1** in s

Note that, by the Gale evenness condition, we must have an the odd maximal run of 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*.

cite: lhg-ref

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	**1**2343**2**	drop **1**
	11110**0**	pivot
	0111**1**0	
pick **3**, duplicate	12**3**43**2**	drop the other **3**
	011**1**10	pivot
	110**1**10	
pick **1**, starting label	**1**23432	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.

edited from paper notes until here

Theorem 1. *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.

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.

□

Theorem 1 holds for odd d as well.

expand - or cut?

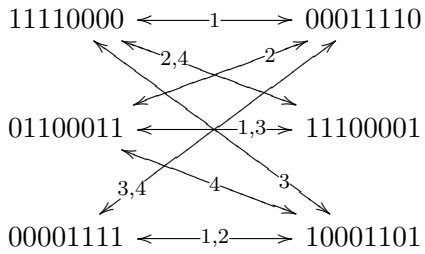
The reversibility of the pivoting steps leads to the following property

proposition 1. *lhg-labels-property 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 **??**.

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 1 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 **??**.

3.2 The Complexity of Gale and Another Gale

file: main-result-subsection

References

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

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

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

[4] 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.

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

[6] 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.

[7] 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.

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

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

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

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

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

[13] 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.

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

better other article, “Exponentially many steps for finding a NE in a bimatrix game.” In the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004.?

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

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

[17] L. Végh, B. von Stengel “Oriented Euler Complexes and Signed Perfect Matchings.” arXiv:1210.4694v2 [cs.DM]

textbooks - papad, ziegler, sth gth