

## Abstract

This thesis presents a report on original research, published as joint work with Merschen and von Stengel in *Electronic Notes in Discrete Mathematics* [1]. 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 introduced by Gale in [5].

Gale strings can be used in the representation of a particular class of games that Savani and von Stengel [10] used as an example of exponential running time for the classical Lemke-Howson algorithm to find a Nash equilibrium of a bimatrix game [7]. 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 [9] 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 Papadimitriou [3] and Chen and Deng [2] proved the PPAD-completeness of finding a Nash equilibrium in general normal-form games, we have a special class of games, unless  $\text{PPAD} = \text{P}$ .

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 [10]. The second one is the polynomial-time solvability of the problem of finding a perfect matching in a graph, proven by Edmonds [4].

Further results by Merschen [6] and Végh and von Stengel [11] will be mentioned.

An appendix relates an amendment to the proof of the PPAD-completeness result by Daskalakis, Goldberg and Papadimitriou [3].

# 1 Introduction

## 2 Complexity, Games, Polytopes and Gale Strings

### 2.1 Some Complexity Classes

reference - take Papadimitriou (book) for general, then article Megiddo and Papadimitriou (1991) for (T)FNP and Papadimitriou 1994 for PPAD

A *computational problem* is given by the combination of an *input* and a related *output*. A specific input gives an *instance* of the problem.

complement of a problem - needed for co-NP

Computational problems can be classified according to the form of their output. A *decision problems* outputs either “YES” or “NO”. An instance *x function problem*, on the other hand, returns a more generic output *y* that satisfies a given relation  $R(x, y)$ .

*Search problems* are function problems that return either an output *y* satisfying a given relation  $R(x, y)$  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*. *Counting problems*, finally, return the *number* of *y*’s that satisfy  $R(x, y)$ ; given a problem *R* we denote the associated counting problem  $\#R$ .

An example of decision problem is: “(input) given a graph, (question) is it possible to find an *Euler tour* for 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.” A counting problem is “(input) given a graph, (output) return the number of its Euler tours.”

Computational problems are also classified according to their *computational complexity*, given by the *reducibility* from each other.

Turing machines: here - not that in the following deterministic TM

Let  $P_1$  be a computational problem. For an instance  $x$  of  $P_1$ , let  $|x|$

be the the number of bits needed to encode  $x$ .  $P_1$  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  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  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.

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. A problem is in the class **co - NP** if its complement is in **NP**. The class **#P** captures the problem of counting the number of possible certificates for a problem in **NP**.

Formally,  $\#P$  is defined as...

In [8]

The class **FNP**, *function non-deterministic polynomial*, is defined as the class of binary relations  $R(x, y)$  such that there is a polynomial-time algorithm that decides whether  $R(x, y)$  holds for given  $x, y$  satisfying  $|y| \leq p(|x|)$ , where  $p$  is a polynomial. If a  $y$  as above is guaranteed to exist, the problem belongs to the class **TFNP**, *total function non-deterministic polynomial*. That is, **FNP** and **TFNP** are analogous to **NP**, but they allow for problems of (respectively) function and total function form.

Cite: Papadimitriou / Megiddo 1991 - def of (T)FNP

More on TFNP: no complete pbls unless NP=co-NP (def co-NP)

⇒

definition of PPA(D)

## 2.2 Normal Form Games and Nash Equilibria

## 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

## 3.2 The Lemke-Howson Algorithm and Parity

file: pivoting-LH-subsection

### **3.3 The Complexity of GALE and ANOTHER GALE**

file: main-result-subsection - done!

## **4 Further results**

**Appendix A: A result about PPAD completeness of Nash**

**Appendix B: Notation**

## References

- [1] M. M. Casetti, J. Merschen, B. von Stengel (2010). “Finding Gale Strings.” *Electronic Notes in Discrete Mathematics* 36, pp. 1065–1082.
- [2] X. Chen, X. Deng (2006). “Settling the Complexity of 2-Player Nash Equilibrium.” *Proc. 47th FOCS*, pp. 261–272.
- [3] 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.
- [4] J. Edmonds (1965). “Paths, Trees, and Flowers.” *Canad. J. Math.* 17, pp. 449–467.
- [5] 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.
- [6] J. Merschen (2012). “Nash Equilibria, Gale Strings, and Perfect Matchings.” PhD Thesis, London School of Economics and Political Science.
- [7] C. E. Lemke, J. T. Howson, Jr. (1964). “Equilibrium Points of Bimatrix Games.” *J. Soc. Indust. Appl. Mathematics* 12, pp. 413–423.
- [8] N. Megiddo, C. H. Papadimitriou (1991). “On Total Functions, Existence Theorems and Computational Complexity.” *Theoretical Computer Science* 81, pp. 317–324.
- [9] 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.
- [10] R. Savani, B. von Stengel (2006). “Hard-to-solve Bimatrix Games.” *Econometrica* 74, pp. 397–420  
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.?
- [11] L. Végh, B. von Stengel “Oriented Euler Complexes and Signed Perfect Matchings.” arXiv:1210.4694v2 [cs.DM]