

# 1 Complexity, Games, Polytopes and Gale Strings

## 1.1 Some Complexity Classes

references - cite Papadimitriou (book) for general; 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.

Computational problems can be classified according to the form of their output. A *function problem*  $P$  returns for an instance  $x$  an output  $y$  that satisfies a given binary relation  $R(x, y)$ . In the case of a *decision problems*,  $y$  is either “YES” or “NO”. The *complement* of a decision problem  $P$  is the problem  $\bar{P}$  that returns “NO” for each instance of  $P$  that returns “YES”, and vice versa.

*Search problems* are function problems that return either an output  $y$  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*. *Counting problems* 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 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.” Finally, 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.

For any class  $C$  of decision problems, the class of all complements of the problems in  $C$  is the complement class  $\text{co-}C$ . 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  $C$ -hard problem in  $C$  is complete for  $C$ .

The complexity class  $\mathbf{P}$  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  $\mathbf{P}$  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  $\mathbf{NP}$ , 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  $xy$ .

This means that 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**.

check formal def of # P (?)

In [7], Megiddo and Papadimitriou introduce 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 whether  $R(x, y)$  holds for given  $x, y$  satisfying  $|y| \leq p(|x|)$ , 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.

In [7], Megiddo and Papadimitriou also prove that, unless  $\mathbf{NP} = \mathbf{co-NP}$ , it’s impossible to find a **TFNP**-complete problem. To circumvent this limitation of **TFNP**, Papadimitriou ([10]) 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*).

definition of PPA(D): one in Papadimitriou 1994 and one in DGP the second w END OF THE LINE, use that one.

as an example, BROUWER, SPERNER (look at Papadimitriou 1994)

## 1.2 Normal Form Games and Nash Equilibria

A *finite normal-form game*  $\Gamma = (P, S, u)$ , where  $S = \times_{p \in P} S_p$  and  $u = \times_{p \in P} u^p$ , and both  $P$  and  $S$  are finite, is a model of a strategic interaction. Each *player*  $p \in P$  chooses a probability distribution  $x^p = (x_1, \dots, x_{|S_p|})$  over a set of *strategies*  $s^p \in S_p$ . Since  $x^p$  is a probability distribution, we have  $x_s^p \geq 0$  and  $\sum_s x_s^p = 1$  for every  $p \in P$  and  $s \in S_p$ . If  $x_s^p = 0$  for every strategy  $s^p$  but  $\bar{s}^p$ , the strategy  $x^p$  is the *pure strategy*  $\bar{s}^p$ ; otherwise,  $x^p$

is a *mixed strategy*. The *strategy profile*  $(x^1, \dots, x^{|P|})$  influences the *payoff*  $u^p : S \rightarrow \mathbb{R}$  of each player  $p$ . For each player  $p$  we denote the set of strategy profiles of all his opponents as  $S_{-p} = \times_{q \neq p} x_q^q$ , and for  $s \in S_{-p}$  we denote (what exactly is this?!) as  $x_s = \prod_{q \neq p} x_{s_q}^q$

A *Nash equilibrium* of a game is a strategy profile in which each player cannot improve his expected payoff by unilaterally changing his strategy. Formally, a Nash equilibrium is a strategy profile  $x$  such that for every  $p \in P$  and every  $s_i^p, s_j^p \in S_p$

$$\sum_{s \in S_{-p}} u^p(s_i^p, s)x_s > \sum_{s \in S_{-p}} u^p(s_j^p, s)x_s \Rightarrow x_j^p = 0$$

The existence of a Nash equilibrium is guaranteed by the following theorem by Nash ([9]).

**Theorem 1.** (Nash, [9]) Every finite game in normal form has a Nash equilibrium.

sketch of pf, brouwer

as example, (generalised) matching pennies, used in DGP, incl appendix here

n-NASH in TFNP and typical problem pushing the def of PPAD ([10]); in fact, PPAD-complete ([3] for  $n \geq 3$ , [2] for  $n = 2$ ).

until here

### 1.3 Best Response Polytopes

file: polytopes

### 1.4 Cyclic Polytopes and Gale Strings

### 1.5 The Problem ANOTHER GALE

file: gale-def

merge in one section "Gale strings" or "CP and GS"?

## 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] C. E. Lemke, J. T. Howson, Jr. (1964). “Equilibrium Points of Bimatrix Games.” *J. Soc. Indust. Appl. Mathematics* 12, pp. 413–423.
- [7] N. Megiddo, C. H. Papadimitriou (1991). “On Total Functions, Existence Theorems and Computational Complexity.” *Theoretical Computer Science* 81, pp. 317–324.
- [8] J. Merschen (2012). “Nash Equilibria, Gale Strings, and Perfect Matchings.” PhD Thesis, London School of Economics and Political Science.
- [9] J. F. Nash (1951). “Noncooperative games.” *Annals of Mathematics*, 54, pp. 289–295.
- [10] 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.
- [11] 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.?
- [12] L. Végh, B. von Stengel “Oriented Euler Complexes and Signed Perfect Matchings.” arXiv:1210.4694v2 [cs.DM]