Complexity of Pure Nash Equilibria in Congestion Games

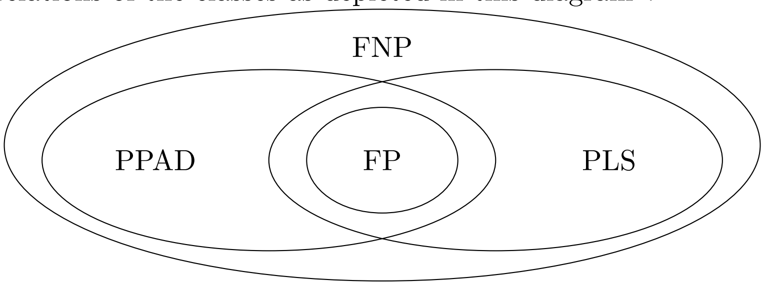
Instructor: Thomas Kesselheim

We have seen that every finite game has a mixed Nash equilibrium. Every congestion game even has a pure Nash equilibrium. However, no algorithm is known to efficiently compute these equilibria. Therefore today's question is: What is the computational complexity of finding an equilibrium? How can one formally show hardness?

1 Complexity Classes for Search Problems

We know the respective equilibrium always exists. Therefore, we are not talking about decision problems here but about *search problems*. This kind of problems is covered by a complexity class called FNP, which extends NP. Stated informally, a search problem belongs to FNP if, given an instance and a solution, one can verify in polynomial time whether this solution is correct. Search problems in FP can be solved in polynomial time, which means that, for a given instance, one can compute a feasible solution in polynomial time or output that there is none.

The mentioned equilibrium problems are contained in FNP. However, they are not FNP-complete. They are typical representatives of two subclasses, called PLS and PPAD. Both PLS and PPAD contain many problems for which no polynomial time algorithm is known. Therefore, PLS- or PPAD-completeness gives strong evidence that a search problem is hard to solve. One can think of the relations of the classes as depicted in this diagram[1]:



The notion of "polynomial time" always depends on the input encoding: What is considered the input size? For example, in a bimatrix game, the input size is the size of the matrices. In this representation, it is easy to find a pure Nash equilibrium in polynomial time because the input length corresponds to the number of pure states and we only have to check each of these states. Finding a mixed Nash equilibrium, however, is PPAD-complete. We will not cover the definition of PPAD.

A congestion game can be represented much more succinctly. One only has to define a delay function for every resource. In case of n players and m resources, these are only nm numbers. However, a single player may have up to 2^m different strategies. Therefore, the number of states can be as large as $(2^m)^n = 2^{nm}$. We will show that finding a pure Nash equilibrium in a congestion game is PLS-complete.

2 The Complexity Class PLS

We will interpret the problem of finding a pure Nash equilibrium as the problem of finding a local optimum. We will show that it is at least as hard as any other local search problem in which single improvement steps work in polynomial time.

[1] Under some assumptions. The relation between PPAD and PLS is, for example, unsettled.

Definition 5.1. *A* local search problem Π *is given by its set of instances* $\mathcal{I}_\Pi$*. For every instance* $I \in \mathcal{I}_\Pi$*, we are given a finite set of* feasible solutions $\mathcal{F}(I)$*, an* objective function $c\colon \mathcal{F}(I) \to \mathbb{Z}$*, and for every feasible solution* $s \in \mathcal{F}(I)$*, a* neighborhood $N(s, I) \subseteq \mathcal{F}(I)$*. A feasible solution* $s \in \mathcal{F}(I)$ *is a* local optimum *if the objective value* $c(s)$ *is at least as good as the objective value* $c(s')$ *of every other feasible solution* $s' \in N(s, I)$*.*

How hard is it to compute a local optimum? To make this a well defined question we need the following definition.

Definition 5.2 (Johnson, Papadimitriou, Yannakakis 1988)**.** *A local search problem* Π *belongs to the class* PLS*, for* Polynomial Local Search*, if the following polynomial-time algorithms exist:*

Algorithm A: Given an instance $I \in \mathcal{I}_\Pi$*, return a feasible solution* $s \in \mathcal{F}(I)$*.*

Algorithm B: Given an instance $I \in \mathcal{I}_\Pi$ *and a feasible solution* $s \in \mathcal{F}(I)$*, return the objective value* $c(s)$*.*

Algorithm C: Given an instance $I \in \mathcal{I}_\Pi$ *and a feasible solution* $s \in \mathcal{F}(I)$*, either certify that* s *is a local optimum or return a solution* $s' \in N(s, I)$ *with better objective value.*

For every problem in PLS one can apply the following natural heuristic:

Local Search Algorithm

1. Use Algorithm A to find a feasible solution $s \in \mathcal{F}(I)$.
2. Iteratively, use Algorithm C to find a better feasible solution $s' \in N(s, I)$ until a locally optimal solution is found.

Note that this local search procedure is guaranteed to terminate because there are only finitely many candidate solutions, and in each iteration the objective function strictly improves. However, because there can be exponentially many feasible solutions, the local search algorithm need not run in polynomial time.

Question: For a given local search problem Π, is there a polynomial-time algorithm (not necessarily local search) for finding a local optimum?

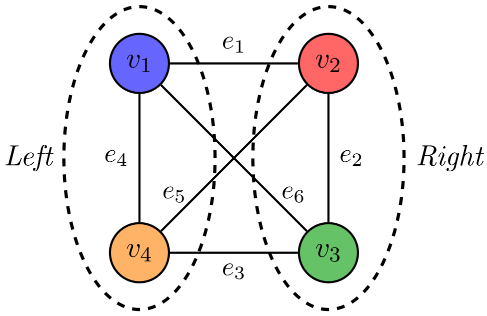
3 Max-Cut and PLS-Completeness

Definition 5.3 (Max-Cut)**.** *The search problem* Max-Cut *is defined as follows.*

Instances:	*Graph* $G = (V, E)$ *with edge weights* $w\colon E \to \mathbb{N}$*.*
Feasible solutions:	*A* cut*, which partitions* V *into two sets* Left *and* Right*.*
Objective function:	*The* value *of a cut is the weighted number of edges with one endpoint in* Left *and one endpoint in* Right*.*
Neighborhood:	*Two cuts are* neighboring *if one can obtain one from the other by moving only one node from* Left *to* Right *or vice versa.*

Observation 5.4. *(Membership in PLS) Max-Cut is a PLS problem.*

Example 5.5. *Consider the following instance of Max-Cut:*



Consider weights $w_{e_i} = i$ for $1 \leq i \leq 6$. Then the cut that separates Left $= \{v_1, v_4\}$ from Right $= \{v_2, v_3\}$ has a weight of 15. The four neighboring cuts each separate one vertex from the other three vertices. These cuts have weight 11, 8, 11, and 12. So the indicated cut is a local optimum. This cut, however, is not a global optimum: the cut that separates v_1, v_2 from v_3, v_4 has weight 17.

Max-Cut unlike Min-Cut is an NP-hard problem, but we are not interested in a globally optimal solution. We only want a local optimum. Intuitively, computing a locally optimal solution should be easier.

A concrete example is Max-Cut in graphs $G = (V, E)$ with weights $w_e = 1$ for all $e \in E$. Here, we can find a local optimum with the local search algorithm in at most $|E|$ steps, while computing a maximum cut remains NP-hard.

Quite surprisingly, for Max-Cut in graphs with general weights no polynomial-time algorithm for computing a local optimum is known; and we can show that this problem is "as hard as" any other local search problem.

Definition 5.6 (PLS-reduction)**.** *Given two PLS problems Π_1 and Π_2, there is a PLS-reduction (written $\Pi_1 \leq_{PLS} \Pi_2$) if there are two polynomial-algorithms f and g:*

- *Algorithm f maps every instance $I \in \mathcal{I}_{\Pi_1}$ to an instance $f(I) \in \mathcal{I}_{\Pi_2}$.*
- *Algorithm g maps every local optimum s of $f(I) \in \mathcal{I}_{\Pi_2}$ to a local optimum $g(s)$ of $I \in \mathcal{I}_{\Pi_1}$.*

As usual, one can read the symbol $\leq_{\text{PLS}}$ as "is not harder than". The reduction $\Pi_1 \leq_{\text{PLS}} \Pi_2$ gives us a way to derive an algorithm for Π_1 from an algorithm for Π_2.

Definition 5.7 (PLS-completeness)**.** *A problem Π^* in PLS is called* PLS-complete *if, for every problem Π in PLS, it holds $\Pi \leq_{PLS} \Pi^*$.*

It is generally assumed that there are problems in PLS that cannot be solved in polynomial time. For this reason, showing PLS-completeness effectively shows that presumably there is no polynomial-time algorithm.

Theorem 5.8 (Schäffer and Yannakakis 1991)**.** *Max-Cut is PLS-complete.*

As in the theory of NP-completeness, showing PLS-completeness requires an "initial" PLS-complete problem. Such a problem was given by Johnson et al.; PLS-completeness of other problems such as Max-Cut can then be established through reduction. It is worth pointing out that in the original problem, local search takes (worst-case) exponential time and that all known reductions preserve these bad instances.

4 PLS-Completeness of Pure Nash in General Congestion Games

Let us now interpret the problem of finding a pure Nash equilibrium in a congestion game as a local search problem. In this case, the feasible solutions are strategy profiles. They are neighboring if they differ in the choice of only a single player. We have seen before that in a unilateral deviation step, the Rosenthal potential changes by the same amount as the respective player's cost. For this reason, pure Nash equilibria correspond to local minima of Rosenthal's potential function.

Observation 5.9 (Membership in PLS)**.** *It is a PLS problem to find a pure Nash equilibrium in a congestion game.*

Theorem 5.10. *Max-Cut $\leq_{PLS}$ Pure Nash Equilibrium in Congestion Games*

Proof. For this reduction, we have to map instances of Max-Cut to congestion games. Given a graph $G = (V, E)$ with edge weights $w \colon E \to \mathbb{N}$, this game is defined as follows. Players correspond to the vertices V. For each edge $e \in E$, we add two resources r_e^{left} and r_e^{right}. The delays are defined by

$$d_{r_e^{\text{left}}}(k) = d_{r_e^{\text{right}}}(k) = \begin{cases} 0 & \text{for } k = 1 \\ w_e & \text{for } k \geq 2 \end{cases} .$$

Each player $v \in V$ has two strategies, namely either to choose all the "left" resources for its incident edges $\{r_e^{\text{left}} \mid v \in e\}$ or all the "right" resources for its incident edges $\{r_e^{\text{right}} \mid v \in e\}$.

This way, cuts in the graphs are in one-to-one correspondence to strategy profiles of the game. A cut of weight W is mapped to a strategy profile S of Rosenthal potential $\sum_{e \text{ not cut}} w_e = \sum_{e \in E} w_e - W$ and vice versa. To see this, consider an edge $e \in E$. If its endpoints are in different sets of the cut, its resources contribute nothing to the potential; if its endpoints are in the same set, then the contribution is $0 + w_e = w_e$.

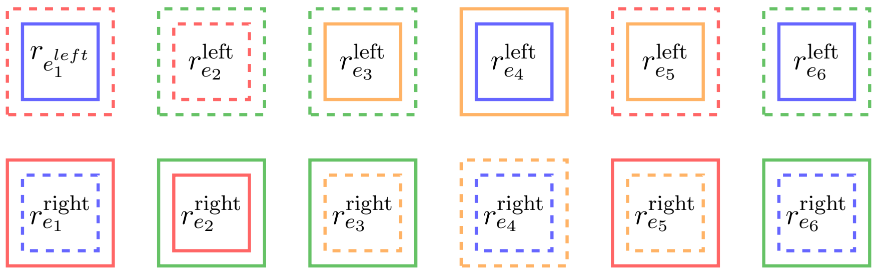


Figure 1: Potential game instance derived from the Max-Cut instance in Example 5.5. Players are color-coded. A player can either choose all solid or all dashed boxes in his color. The solid boxes correspond to the locally optimal cut that separates v_1, v_3 from v_2, v_4. The only non-zero contributions to the potential function come from the edges that are contained in either Left or Right, namely e_2 and e_4 with weight 2 and 4. Their sum, 6, is equal to the total edge weight minus the weight of the cut, $21 - 15$.

Consequently, local maxima of Max-Cut correspond to local minima of the Rosenthal potential, which are exactly the pure Nash equilibria. Therefore, the second part of the reduction is again trivial. $\square$

Note that the above reduction was to a congestion game in which the players' strategy sets are not identical (so it is asymmetric) and required strategies to be arbitrary subsets of the resources (as opposed to, e.g., paths in a network). It turns out that either restriction can be dropped and the problem remains PLS-complete; but dropping both turns the problem into one that one can solve in polynomial time.

Recommended Literature

- D. S. Johnson, C. H. Papadimitriou, M. Yannakakis. How easy is local search? Journal of Computer and System Sciences, 37(1):79-100, 1988. (Class PLS, Cook-Like Theorem for CircuitFlip)
- A. Fabrikant, C. H. Papadimitriou, K. Talwar. The complexity of pure Nash equilibria. STOC 2004. (First Proof of PLS-Completeness of Pure Nash in Congestion Games)
- A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. SIAM Journal on Computing, 20(1):56-87, 1991. (PLS-Completeness in Congestion Games via Max-Cut)
- H. Ackermann, H. Röglin, B. Vöcking. On the impact of combinatorial structure on congestion games. Journal of the ACM, 55(6), 2008. (Further Results on PLS-Completeness)