Algorithmic Game Theory, Summer 2015 Lecture 3 (4 pages)

Complexity of 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?

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.

The mentioned equilibrium problems are contained in FNP. However, they are not FNP-complete. Therefore, one has focused attention on subclasses of FNP, namely PLS and PPAD. Both PLS and PPAD contain 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.

Finding a mixed Nash equilibrium in a finite game is PPAD-complete. Even understanding the definition of PPAD and seeing that this problem is contained to PPAD is fairly complex. Therefore, we will instead consider pure Nash equilibria in congestion games.

1 The Complexity Class PLS

Rosenthal's potential function allows us to interpret congestion games as local search problems: Nash equilibria are local optima with respect to potential function.

Therefore the question boils down to: How difficult is it to compute local optima?

Definition 3.1. *The complexity class PLS (Polynomial Local Search) contains search problems with an* objective function *and a specified* neighborhood relationship Γ. *It is required that there are polynomial-time algorithms A, B, and C for the following tasks.*

- *Algorithm A computes an arbitrary feasible solution (the objective function value is irrelevant).*
- *Given any solution s, algorithm B computes its objective value.*
- *Given any solution s, algorithm C either*
  - *computes a solution in $\Gamma(s)$ with better objective value, or*
  - *certifies that s is a local optimum.*

Definition 3.2. *The search problem* Positive Not-All-Equal kSat *(Pos-NAE-kSAT) is defined as follows.*

Input: *Formula over n binary variables $x_1, \dots, x_n$ described by m clauses $c_1, \dots, c_m$ each containing k positive literals, and m weights $w_1, \dots, w_m$.*

Feasible Solutions: *Any assignment $s \in \{0,1\}^n$*

Objective Function: *Clause c_i is* satisfied *by an assignment $s \in \{0,1\}^n$ if its literals are not all assigned the same value. The* value *of an assignment is the weighted sum of satisfied clauses.*

Neighborhood Relationship: *Assignments s and s' are* neighboring *if they differ in exactly one position.*

Note that, as a local search problem, one has to only find a *local optimum*, i.e., an assignment without neighboring assignment of higher value.

Example 3.3. *Example instance of Pos-NAE-3SAT:*

$$c_1 = x_1x_2x_3;\ c_2 = x_1x_2x_4;\ c_3 = x_1x_2x_5;\ c_4 = x_3x_4x_5$$

$$w_1 = 100;\ w_2 = 110;\ w_3 = 120;\ w_4 = 100$$

An example for a local optimum *is the assignment*

$$x_1 = 0;\ x_2 = 0;\ x_3 = 1;\ x_4 = 1;\ x_5 = 1$$

of value 330. Each of the five neighboring assignments has a value of at most 330.

Definition 3.4 (Max-Cut)**.** *The search problem* Max-Cut *is defined as follows.*

Input: *Graph $G = (V, E)$ with edge weights $w : E \to \mathbb{N}$.*

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 Relationship: *Two cuts are* neighboring *if one can obtain one from the other by moving only one node from* Left *to* Right *or vice versa.*

Definition 3.5 (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 A and B:*

- *Algorithm A maps every instance x of Π_1 to an instances of Π_2.*
- *Algorithm B maps every local optimum of $A(x)$ to a local optimum of x.*

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.

Theorem 3.6. *Pos-NAE-3Sat $\leq_{PLS}$ Pos-NAE-2Sat*

Proof. For each 3-clause (x_1, x_2, x_3) of weight w, Algorithm A introduces the three 2-clauses $(x_1, x_2), (x_1, x_3), (x_2, x_3)$ each of weight $w/2$.

The value of an assignment in the 2SAT instance is identical to its value in the 3SAT instance as a 3-clause is satisfied if and only if exactly two of the three corresponding 2-clauses are satisfied.

Hence, the local optima of both instances coincide (Algorithm B is trivial), and the conditions of a PLS-reduction are fulfilled. □

Theorem 3.7. *Pos-NAE-2Sat $\leq_{PLS}$ Max-Cut*

Proof. Each variable is represented by a node. Each clause and its weight is represented by a weighted edge. Multi-edges are merged to single edges by adding their weight.

Given an assignment for the variables, 0 is interpreted as *Left* and 1 is interpreted as *Right*. This way, the local optima of both instances coincide, and the conditions of a PLS-reduction are fulfilled. □

Definition 3.8 (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. (As PLS is only a subclass of FNP, this is a stronger statement than P $\neq$ NP.) For this reason, showing PLS-completeness effectively shows that presumably there is no polynomial-time algorithm. One can show that POS-NAE-3SAT is PLS-complete in a similar way as one shows that SAT is NP-complete. By transitivity of $\leq_{\text{PLS}}$, we immediately get the following corollary.

Corollary 3.9. *POS-NAE-2SAT and Max-Cut are PLS-complete.*

2 Complexity of Pure Nash Equilibria in Congestion Games

Let us now interpret the problem of finding a pure Nash equilibrium in a congestion game as a 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 3.10. *It is a PLS problem to find a pure Nash equilibrium in a congestion game.*

For the problem of computing a pure Nash equilibrium in congestion games, Fabrikant, Papadimitriou, and Talwar show the following results.

	network games	general games
symmetric	$\exists$ poly-time algo	PLS-complete
asymmetric	PLS-complete	PLS-complete

We present only one of these PLS-completeness proof, the one for general, asymmetric congestion games (the simplest one).

Theorem 3.11. *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 of Rosenthal potential $\sum_{e \in E} w_e - W$ 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 $0 + w_e = w_e$.

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. □

In a symmetric network congestion game, there is a directed graph $G = (V, E)$ with delay functions $d_e \colon \{1, \ldots, n\} \to \mathbb{Z}$, $e \in E$ with a source $s \in V$ and a target $t \in V$, such that each player wants to allocate a path of minimal delay between s and t. In more general *asymmetric* network congestion games, different players might have different source-destination pairs.

It is known that there are instances of symmetric congestion games in which there are states such that every improvement sequence from this state to a Nash equilibrium has exponential length. Hence, applying improvement steps is not an *efficient* (i.e. polynomial time) algorithm for computing Nash equilibria in these games. However, there is another algorithm which finds Nash equilibria in polynomial time.

Recommended Literature

- Tim Roughgarden's lecture notes `http://theory.stanford.edu/~tim/f13/l/l19.pdf` and lecture video `https://youtu.be/5xu8C0wj7UU`
- M. Yannakakis. Computational complexity. Chapter 2 in E. Aarts and J. Lenstra (Eds), "Local Search in Combinatorial Optimization", pages 19–55, 1997. (Survey of the class PLS)
- A. Fabrikant, C. Papadimitriou, K. Talwar. The complexity of pure Nash equilibria. STOC 2004. (PLS-Completeness in Congestion Games)
- 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)