CS295 Introduction to Algorithmic Game Theory

Instructor: Ioannis Panageas

Scribed by: Divyaanand Sinha, Fivos Kalogiannis

Lecture 9. PLS and PPAD reductions.

1 Reductions in NP

We first define and look into some of the well-known problems in NP and then discuss how one can be reduced to another.

Definition 1.1 INDEPENDENT-SET *(IS) problem: Given an undirected graph $G = (V, E)$, is there an* INDEPENDENT-SET *of vertices $I \subset V$ such that the vertices in I are not connected pairwise by an edge in E.*

Definition 1.2 3-SAT *problem: Given a boolean expression of 3 literals in Conjunctive Normal Form (CNF), is there an assignment to the literals such that the expression evaluates to true.*

1. A **literal** is basically a boolean variable or its negation. For example if x is a boolean variable then both x and $\neg x$ are literals.
2. A boolean expression is said to be in **Conjunctive Normal Form** if it is constructed by a conjunction of **Clauses** for example $C_1 \wedge C_2 \wedge C_3$.
3. A **Clause** is a boolean expression constructed by a disjunction of **literals** for example $x \vee y \vee \neg z$.

One example of a 3-SAT problem is the following boolean expression

$$E = (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \tag{1}$$

One possible assignment of variables is $x = 1$, $y = 1$, $z = 0$ to make the expression E evaluate to 1. On the other hand, the following expression

$$E = (x \vee \neg y \vee \neg y) \wedge (\neg x \vee \neg y \vee \neg y) \wedge (x \vee y \vee y) \wedge (\neg x \vee y \vee y) \tag{2}$$

cannot be satisfied for any possible assignment of the variable. The 3-SAT problem is NP-complete.

1.1 3-SAT reduction to IS

2 Complexity of Pure Nash Equilibria (PNE) in Congestion Games

In this section we discuss the complexity of finding Pure Nash Equilibria in Congestion Games. We first inrtoduce the class PLSand show that computing Pure Nash Equilibria in congestion games is PLS-complete.

2.1 The class PLS

PLS(Polynomial Local Search) is class for local search algorithms. A problem in PLS is specified by the following three algorithms.

1. The algorithm takes as input an instance and outputs an arbitrary feasible solution (not necessarily optimal).

2. An algorithm/oracle which takes as input a feasible solution and computes the objective function value c of the solution. The objective function is the one we want to minimize.

3. An algorithm which takes as input the feasible solution and computes another feasible solution with objective function value c' such that $c' < c$ or None if such a solution doesn't exist.

The LOCAL MAX-CUT problem belongs to the class PLS. Given a graph $G = (V, E)$ the goal of LOCAL MAX-CUT is to divide the vertices into X and $V - X$, such that sum of the weights of all the edges crossing the two cuts is a local maximum. A local maximum implies that the cut weight cannot be increased by switching a vertex from one set to another. We can observe that for a graph the LOCAL MAX-CUT can be computed the set of three algorithms defined earlier. Specifically, given a graph G we first compute an arbitrary cut of the graph given by the sets X and $V - X$. We then check the sum of the edge weights in the weight. Finally, we try to see if we can increase the cut weight by switching any vertex v from one set to another, if possible we switch it. By applying these steps iteratively, ultimately we arrive at a cut for the graph which is locally maximum.

Theorem 2.1 *The* LOCAL MAX-CUT *problem is* PLS*-complete.*

2.2 Computing PNE in congestion games $\in$ PLS

We show that computing PNE in congestion games is an instance of PLS by first describing the three algorithm needed for PLS as follows:

1. Take as input an instance of a congestion games consisting of players, bins/edges and output an arbitrary strategy profile s.

2. Take as input the congestion game and s and return the objective function computed as $\Phi(s) = \sum_e \sum_{j=1}^{j=l_e(s)} c_e(j)$.

3. Check if the given strategy profile is PNE by checking if we can switch the strategy of agent i from s_i to s_i' such that $\Phi(s'(i); s_{-i}) < \Phi(s'(i); s_i)$. In other words if changing the strategy of any one of the agents reduces the potential function.

All of these algorithms run in polynomial time with respect to the description of the game i.e. number of agents, edges etc.

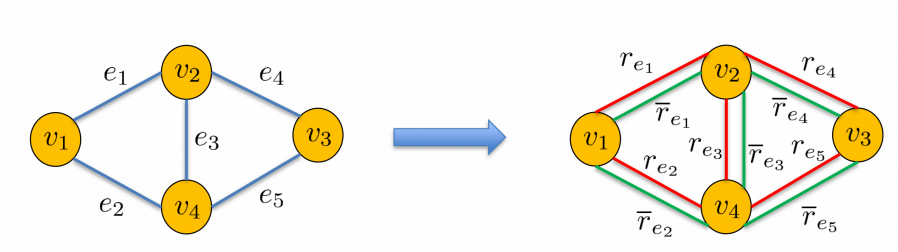

Figure 1: Tranformation of a graph to congestion game.

2.3 Reducing LOCAL MAX-CUT TO PNE in congestion games

We will now show that finding the LOCAL MAX-CUT problem can be reduced to finding PNE in Congestion Games.
Given a weighted graph $G = (V, E)$ we define the following congestion game.

1. Agents are the vertices in V.
2. For each edge $e \in E$ we have two resources r_e and $\bar{r}_e$
3. Each agent v has two strategies $s_v = \{r_e : \forall e \text{ incident to } v\}$ or $\bar{s}_v = \{\bar{r}_e : \forall e \text{ incident to } v\}$.
4. The cost of using a resource $r_e/\bar{r}_e$ is 0 if one player uses it otherwise w_e if two players use it.

Note that the above transformation can again be done in polynomial time in the description of the graph. The transformation is shown in Figure 2.3 for easier visualization. At the end of the transformation we can separate the vertices into two sets, ones that chose r_e and the others that chose $\bar{r}_e$. This also represents a cut of the original graph. Therefore, given a cut in this graph the sum of the edge weights crossing the cut can be written as the following.

$$w(X, V - X) = \sum_{e=(u,v):u\in X, v\in V-X} w_e \tag{3}$$

The above summation can also be calculated by summing up the edge weights of all the edges in the graph, and subtracting the edge weights which are within the set X and $V - X$ as the following.

$$w(X, V - X) = \sum_{e\in E} w_e - \sum_{e=(u,v):u\in X, v\in X} w_e - \sum_{e=(u,v):u\in V-X, v\in V-X} w_e \tag{4}$$

Note that in the last two summations for the edges inside the sets X and $V - X$ we always have cost w_e since two agents are always using it. For the edges crossing the cut the cost is zero since only one agent is using it. Hence the last two summations in the above equation are basically the potential function of the congestion game and the equation can be re-written as the following

$$w(X, V - X) = \sum_{e \in E} w_e - \Phi(X, V - X). \tag{5}$$

Therefore computing the LOCAL MAX-CUT is equivalent to minimizing the potential of the congestion game.

Theorem 2.2 *Computing in Pure Nash Equilibria in Congestion Games is* PLS*-complete.*

3 The complexity of computing Nash equilibria

3.1 Total search problems

The *search problem* counterpart of the NP complexity-class is the FNP complexity class — *i.e.,* we are not only interested in whether a solution to a given problem exists, but we also want to compute it. The problem of Nash equilibrium computation in multi-player normal-form games comes in sharp contrast with numerous problems in FNP; we are always guaranteed that they exist thanks to the theorem of the existence of Nash equilibria in finite games. Problems like this are said to be *total* — we know that a solution exists for any instance of the problem *a priori*. The complexity class that contains *total* search problems that belong in NP is known as TFNP.

The TFNP complexity class contains numerous complexity classes that are characterized by the argument used to conclude the existence of a solution to their corresponding family of problems. For example, for the existence of Nash equilibria we used the *Brouwer fixed point theorem*. Further, the problem PIGEON is known to always have a solution thanks to the *pigeonhole principle*; the corresponding complexity class for which PIEGON is complete is PPP.

Namely, we list the following classes that belong to TFNP with a **rough** description:

- PPP-(Polynomial pigeonhole principle) The class of total problems whose solution existence is guaranteed by the pigeonhole principle.
- PPA-(Polynomial parity argument) The class of total problems for which the *handshake lemma* guarantees the existence of solutions.
- PPAD-(Polynomial parity arguments on directed graphs) The complexity class whose problems are guaranteed to have a solution thanks to the Sperner's lemma (or Brouwer's fixed point theorem).
- PLS-(Polynomial local search) The set of problems where the goal is to find a local optimum of a given function.

3.2 The complexity class PPAD

In order to formally define the complexity class PPAD we need to first define the END-OF-LINE problem:

Definition 3.1 (END-OF-LINE) *Given two circuits $P(\cdot)$ and $S(\cdot)$ each with m input bits and m output bits, such that $P(0^m) = 0^m \neq S(0^m)$, find an input $x \in \{0,1\}^m$ such that $P(S(x)) \neq x$ or $S(P(x)) \neq x \neq 0^m$.*

In words, there exists an underlying directed graph G of maximum in and out degree equal to 1. The graph G has size exponential in m. One special vertex, 0^m, is set to be its own predecessor. The two circuits can efficiently (*i.e.,* in time polynomial in m) return the predecessor and successor of any given vertex. Our goal is to find a vertex that has no successors (*i.e.*, lies at the end of a line) or has no predecessor and is not 0^m.

The problem END-OF-LINE is complete for PPAD and it is further used to prove hardness of relevant problems. In fact, in order to prove the hardness of computing a Nash equilibrium in normal-form games we reduce the hardness of computing a Brouwer fixed point to the problem of END-OF-LINE.

3.3 The 2D Sperner's Lemma

Theorem 3.1 *Consider the two-dimensional simplex Δ^3 and a triangulation of it. Further, let a proper 3-coloring of the triangulation; every vertex of the original triangle is assigned a unique color, the vertices on the original edges are colored with only the two colors of the edge's ends, and interior vertices are colored arbitrarily. Then, there always exists an odd number of trichromatic triangles.*

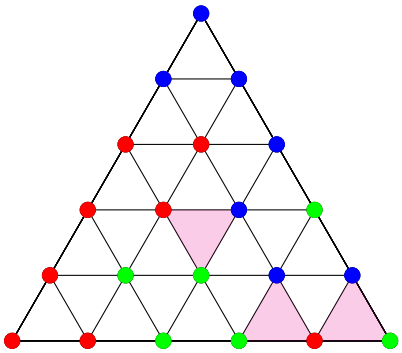

Figure 2: A proper Sperner triangulation

The computational problem that corresponds to finding a panchromatic triangle is known as 2D-SPERNER. In fact, for any proper triangulation we can implicitly generate a directed acylcic graph with maximum in- and out-degrees equal to 1.

Proof: We add a boundary to the initial triangle in such a way that we rule out the possibility of ever leaving the triangle by traversing it with the rule we will use. In particular, we add and

extra triangle as the one shown in 3. Create an appropriate triangulation by drawing edges between the boundary parallel edges.

Select two colors arbitrarily, *e.g.*, red and green – we will use them to define a rule of traversing the triangle in such a way that we always end up in a panchromatic triangle. When in a non-panchromatic A triangle, move to the neighboring triangle B such that the edge shared by A and B is red on the left and green on the right as seen when moving from the centre of A to that of B.

If we cannot leave a triangle this can only mean that we are in a panchromatic triangle as the remaining vertex (other than the one we crossed to enter) has to be a color different than red and green.

Further, the possibility of forming a circular trajectory inside the triangle is ruled out thanks to our rule.

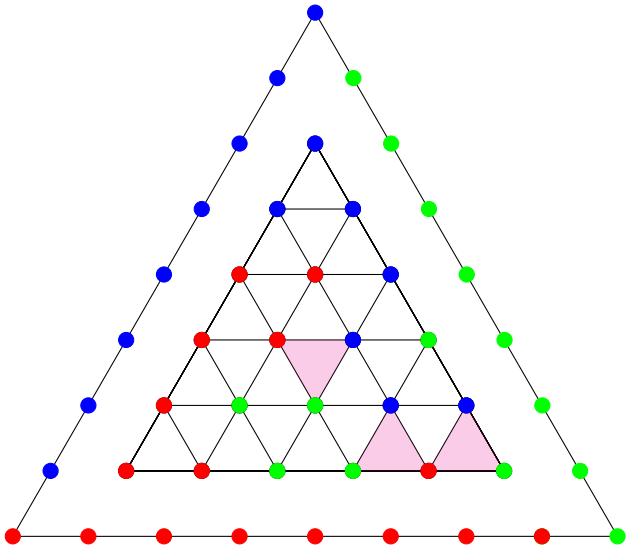

Figure 3: Adding a boundary

■

The problem of computing a Brouwer fixed point is known as BROUWER. Formally it is defined as follows,

Definition 3.2 (BROUWER) *Let a poly-time algorithm Π_F that computes the value of function $F : E^n \to [0,1]^n$ that is K-Lipschitz. Also, let a scalar $\epsilon > 0$. Find a $x \in [0,1]^n$ such that: $\|f(x) - x\|_\infty \leq \epsilon$.*

A higher dimensional version of the SPERNER problem is not much unlike the END-OF-LINE problem.

3.4 The complexity of computing a Nash equilibrium

Consider the function used in the proof of NE existence:

$$f_{i,s} = \frac{x_i(s) + \max\{0, u_i(s, x_{-i}) - u_i(x)\}}{1 + \sum_{s' \in S_i} \max\{0, u_i(s, x_{-i}) - u_i(x)\}}. \tag{6}$$

The function is known to attain a fixed point as is shown in the proof of Nash's theorem. This way, we can reduce the problem of finding a Nash equilibrium to finding a Brouwer fixed point.

Further, for the computational problem of computing an ϵ-approximate Nash equilibrium it suffices to have a circuit that simulates the vector function f in polynomial time. *I.e.,* the problem of finding an approximate Nash equilibrium reduces to computing a Brouwer fixed point. In turn, a Brouwer fixed point is equivalent to the problem of END-OF-LINE.

Moreover, Daskalakis and Papadimitriou [2005] and Chen and Deng [2005] showed that BROUWER can be reduced to computing an approximate equilibrium of 2-player general-sum games proving the PPAD-completeness of the problem of computing Nash equilibria.

References

Constantinos Daskalakis and Christos H Papadimitriou. Three-player games are hard. In *Electronic colloquium on computational complexity*, volume 139, pages 81–87. Citeseer, 2005.

X Chen and X Deng. Settling the complexity of 2-player nash equilibrium. eccc. Technical report, Report, 2005.