Lecture 4

Lecturer: Noam Nisan *Scribe: Shahar Dobzinski*

1 Reminder

Let's recall a few basic definitions. A *game* consists of n players, each having a set of strategies S_i. We define the utility of the i'th player to be $u_i : S_1 \times ... \times S_n \to \mathbb{R}$. We say that a strategy $s_i \in S_i$ is a *best response* to s_{-i}, if for all $s'_i \in S_i$, $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$. A *Nash equilibrium* is a vector of strategies $s_1, ..., s_n$, where for each i $s_i \in S_i$, s_i is a best response to s_{-i}. Observe that they may exist exponential number (in the number of strategies) of Nash equilibria.

2 The MAX-CUT game

In this section we will consider one particular game, the MAX-CUT game. The game is completely defined by a graph $G = (V, E)$. We identify each player with a vertex $v \in V$. Each player has exactly 2 strategies, black or white. We let the utility function of each player to be $u_i(s_1, ..., s_n) = |\{u|e \in (u, v), s_i \neq s_v\}|$. If we are also given weights for each edge, we can define the weighted version of the game in a similar vein.

Lemma 1 *For every graph G the corresponding game has a pure Nash equilibrium.*

Proof: We give two proofs for this lemma.

1. Observe that the optimal solution of the MAX-CUT defines a Nash equilibrium. Define all strategies of the players to be according to the cut. That is, let all players which their corresponding vertices are in one side of the cut to play black, and all the other vertices to play white. Now suppose that there exists a player which his best response is to switch strategies. Clearly, his benefit from switching sides of the cut is identical to the improvement in the value of the cut. This is a contradiction to the optimality of the cut.

2. We will use a well known 2-approximation algorithm for finding the maximum cut. We will then see that the cut it provides actually defines a Nash equilibrium.

   The Algorithm:

(a) Start with an arbitrary partition of the vertices into two sets.

(b) If there exists a vertex with more neighbours in the other set, move this vertex to the other set. Repeat this step until no such vertex exists.

Claim 2 *The algorithm provides a cut that is a 2-approximation to the optimal cut in polynomial time (in the unweighted version).*

Proof: First observe that the algorithm indeed stops in polynomial time: in every step we improve the value of the solution in at least 1. A trivial upper bound to the value of the cut is $|E|$. Thus, the algorithm must finish in within $|E|$ steps.

As for the approximation ratio, we note that in the end of the algorithm, for each vertex the number of neighbours in the other side of the cut is bigger than the number of neighbours in his side. Clearly, at least $\frac{|E|}{2}$ of the edges are in the cut. Using the upper bound of $|E|$ we conclude that we have a 2-approximation algorithm. ■

We note that for the weighted case the algorithm is polynomial in the value of the largest edge. That is, the algorithm is pseudo-polynomial.

Claim 3 *The cut obtained by the algorithm defines a Nash equilibrium.*

Proof: The claim is clearly true because the algorithm stops only if the best response of each player is his current strategy. ■

An obvious different between the two proofs is the obtaining the first equilibrium point is NP-hard, while the second can be computed in polynomial time. ■

This algorithm uses *local search.* In other words, given a solution we look in its "neighborhood" for another solution with better value. If no such exist, we have found a local maximum. More formally the complexity class PLS (abbreviation of polynomial local search) is defined as follows: given a problem of size n, a set of solutions, and a polynomial time computable function f that provides us with a neighbouring solution with a better value, the problem is to find a solution that is a local optimum. This class is believed to be harder than P. For example, the weighted version of MAX-CUT is complete for this class. Other examples for algorithms in this class are the Simplex and simulated annealing.

3 Potential Games

In this section we will generalize the ideas of the previous section and define the notion of *Potential Game.*

Definition 4 *A game is called a potential game if there exists a function* $P : S_1 \times ... \times S_n \to \mathbb{R}$*, such that for each* i, s_i, s_{-i}, y_i*,* $u_i(s_i, s_{-i}) > u_i(y_i, s_{-i}) \iff P(s_i, s_{-i}) > P(y_i, s_{-i})$*.*

We say that game is exact *if the following holds:*

$$u_i(s_i, s_{-i}) - u_i(y_i, s_{-i}) = P(s_i, s_{-i}) - P(y_i, s_{-i})$$

The following is the main theorem regarding potential games.

Theorem 5 *For every potential game there is a Nash equilibrium in pure strategies.*

We will now consider a few examples.

Example 1 *Battle of the sexes.*

$2,1$	$\leftarrow$ $0,0$ $\downarrow$
$\uparrow$ $0,0$ $\rightarrow$	$1,2$

The arrows denote possible (local) improvements that can be made by the players. Here we have two possible Nash equilibria which may be achieved by the local improvement algorithm.

Example 2 *Prisoner's Dilemma*

$\downarrow$ $3,3$ $\rightarrow$	$0,5$ $\downarrow$
$5,0$ $\rightarrow$	$1,1$

Here we have two Nash equilibria, but only one of them can be achieved by the local improvement algorithm.

Example 3 *Odd/Even*

$\downarrow$ $-1,1$	$\leftarrow$ $1,-1$
$1,-1$ $\rightarrow$	$-1,1$ $\uparrow$

Here there is no Nash equilibrium in pure strategies. Therefore, there is no potential function for this game.

These examples essentially illustrate the next lemma.

Lemma 6 *A game is a potential game if and only if the local improvement always finishes.*

Proof: Define a graph with a vertex for each possible combination of strategies. Add an edge to another combination of strategies if and only if the new combination is a best response for one of the players. A potential function exists if and only if the graph does not contain a circle. We can use topological sort to explicitly build the potential function. ■

We stress that the potential function does not always have an interesting *semantical* meaning, like in the MAX-CUT example. Usually, the potential game is only a syntactic tool to prove the existence of an equilibrium in pure strategies.

4 The Load Balancing Game

Here is another example for a potential game. In this game we have n players, each having a value $w_1, .., w_n$. We think of the players as jobs. The strategies of the players are a set of processors. We define the cost of a player to be the sum of weights of the jobs in his processor (that is, the total weight of players that play the same strategy).

Theorem 7 *The Load Balancing game is a potential game.*

Proof: We start by defining an order relation on two *ordered* n-tuples $(a_1, ..., a_n), (b_1, ..., b_n)$. We assume that the n-tuples are ordered from the largest to the smallest. $(a_1, ..., a_n) > (b_1, ..., b_n)$ if i is the smallest index where $a_i \neq b_i$ and $a_i > b_i$.

The theorem clearly follows by using the order we have just defined on the n-tuples that represent the load on the machines. ■

Unfortunately, the same proof does not work when considering the weighted version of the problem (when each job has a different weight on each machine). However, in this case there exist a different potential function (the sum of the squares of the loads).