CS 684 Algorithmic Game Theory
Lecturer: Mark Sandler **March 12, 2004**

1 Graph representation of multiplayer games

Usually n-player games are represented in tabular form. Say if each agent has t actions available, then the game is given by n, n-dimensional matrices, each of size t^n. Matrix i specify the payoff of player i, given the joint actions of all players. Obviously this approach is not very compact, especially if each player payoff depends only on the actions of the small subset of players.

In this lecture we limit ourselves to two-action games, with maximal payoff of 1[1].

Now we describe graph model for multiplayer games. We represent our n-player game as a pair $(G, \mathcal{M})$, where G is unidirected graph with n nodes, where there exist edge between i and j if and only if actions of player i influence actions of player j. The second element $\mathcal{M} = \{\mathcal{M}_1, \ldots \mathcal{M}_i\}$, represent payoff matrices for each player. Let $N_G(i) \subseteq \{1, \ldots, n\}$ be a set of neighbours of i in G, by definition we include i in $N_G(i)$. Then each $\mathcal{M}_i$ will have dimension $|N_G(i)|$ and hence $2^{|N_G(i)|}$ elements. Say, we bound maximal degree of G by k, then total input size for the game would be order of $O(n2^k)$, the goal is to find NE in time polynomial of the input size. Note that if graph G is a clique, then it represent general case n-player game. So one would expect that problem to be hard for general graphs. Unfortunately even for graph with bounded tree width it is still an open problem, however we can do it for trees.

Even for trees the problem is not trivial, in today's lecture we will present algorithm from the paper [1], which runs polynomial (or quasipolynomial[2]) time and finds all approximate Nash Equilibria (to be defined later) for arbitrary tree. And secondly (if time permits), i will sketch the modification of the algorithm which would allow to find *all* exact equilibria, in exponential time[3]. Natural example for such a game would be a company, where payoff of everyone depends from the person himselves, upper manager actions and action of his subordinates. While actions of CEO influence the payoff of everyone, they do so only by propagating action changes downstream the tree.

2 More Notation

Suppose player i has k neighbors (including himself) and $\vec{x} \in \{0,1\}^k$, $M_i(\vec{x})$ denotes player i payoff when his k neighbors played $\vec{x}$. Analagously for mixed strategy $\vec{p} \in [0,1]^k$, $M_i(\vec{p})$ denotes expected playoff of player i.

For arbitrary vector $\vec{x}$, $\vec{x}[i : p_i]$ denotes vector which is the same as $\vec{x}$, except that i-th coordinate is replaced by p_i.

Definition 1 (Approximate Nash) *We say that set of strategies x defines an approximate nash, with respect to constant ε if no player can benefit by more than ε. In other words for every i and p_i, $\mathcal{M}_i(\vec{x}[i : p_i]) \leq M_i(\vec{x}) + \varepsilon$.*

[1] In general our results would be linearly dependent on maximal payoff

[2] e.g. exp(polylog)

[3] Note, that even this result is not trivial, because no finite algorithm is known for general n-player games.

Since there is correspondence between players and nodes, for simplicity we will also denote by M_V, the payoff matrix of player corresponding to V.

3 Abstract Tree Algorithm

We first present algorithm without specifying data structures (and without even showing that they have finite representation), and then instantiate these structures for approximation algorithm.

First of all since G is a tree, we can root it. Fix some root R. Now every node U (except root) we have uniquely defined parent $P(U)$ and possibly many children. Let G^U denotes a subtree induced by U, and M^U denotes a collection of payoff matrices with players from G^U. If $v \in [0,1]$ denotes some mixed strategy, then let M^U_v denotes the collection of payoff matrices for G^U, except that M_U will be collapsed by one index by fixing the parent of U to play strategy v. By definition we assume that $M^R_v = M^R$.

The idea of the algorithm is to build up a table $t_U(v,u)$ for every node U with the following semantics

$$t_U(v,u) = \begin{cases} 1 \text{ if there is NE in the game } (G^U, \mathcal{M}^U_v), \text{ where } U \text{ plays } u \\ 0 \text{ otherwise} \end{cases}$$

Since mixed strategies form a continuous set, it is not clear how we can represent such a table. However for now we assume that we can do this. Then we can easily build the table starting from the leaves and going up the tree. After we build it we can go back down and reconstruct the Nash Equilibria

Now we present the algorithm to build up tables t_U for all nodes.

Algorithm 1 *Tree Nash.*

1 *Root the tree.*

2 *Order nodes starting from the leaves, so that each node is listed only after all its children are listed. (Say in reverse depth-first order).*

3 *For every node U in t:*

   (a) *(U is a leaf.) For every pair of strategies (v,u), set $t_U(v,u)$ to 1 iff u is the best response for U if the parent of U plays v.*

   (b) *(U is an internal node.) Let $\{U_1, \dots U_k\}$ be the children of U. For all $v, u \in [0,1]$ and all joint mixed strategies $\vec{u} = \{u_1, \dots, u_k\}$, we set $t_U(v,u)$ to be 1, iff $t_{U_i}(u,u_i) = 1$ for all i, and $U = u$ is the best response in the game $(G^U, \mathcal{M}^U_v)$ where children of U are playing $\vec{u}$.*

And to print Nash Equilibrium we just use the following recursive algorithm.

1 Choose a strategy r for root R, such that $t_R(0,r) = 1$.[4]

2 For every node U, with already chosen strategy u, and its parent playing v, we select joint strategy $\vec{u} = \{u_1, \dots u_k\}$ for all its children $\{U_1, \dots U_k\}$ such that:

[4]Note that by definition of $t_R(v,u)$ for root node R, the value of parent strategy v is ignored.

(a) $t_{U_i}(u, u_i) = 1$ for all $1 \le i \le k$.

(b) $U = u$ is the best response in $(G^U, \mathcal{M}_v^U)$, where $\{U_1, \dots U_k\}$ are playing $\{u_1, \dots, u_k\}$.

By construction of t_U we can alway choose $\vec{u}$.

Note, that by chosing different joint stategies $\vec{u}$ satisfying (a) and (b) we can build all possible NE, and step (3) of the construction algorithm ensures that we can always chose feasible $\vec{u}$ and hence we will have NE at the end.

4 Approximation Algorithm

In this section we use algorithm 1, but we consider only discretized strategies with step τ, and then prove that by taking τ small enough we can achieve ε eqiulibria. (Of course we will have to argue that our algorithm stays quasipolynomial)

We start with the following simple lemma:

Lemma 1 *Let $\vec{p}, \vec{q} \in [0,1]^k$ satisfy $\max_i |p_i - q_i| \le \tau$, str then provided $\tau \le 1/k$,*

$$\prod p_i - \prod q_i \le 2k\tau$$

Proof. Indeed, consider the maximal possible attainable difference between $\prod p_i$ and $\prod q_i$. Without loss of generality we shall assume that $\prod p_i > \prod q_i$, but then for all i, we have $p_i - q_i = \tau$ (otherwise we could have made the difference larger), therefore we have:

$$\prod p_i - \prod q_i = \prod (q_i + \tau) - \prod q_i \le \sum_i C_k^i \tau^i \le \sum \frac{k^i \tau^i}{i!} \le 2k\tau$$

where the first inequality comes from the fact that $q_i \le 1$, second from upper bound on C_k^i and third from lower bounding $i!$ by 2^i and the fact that $\tau \le 1/k$. ■

Lemma 2 *Let the mixed strategies $\vec{p}$ and $\vec{q}$ for $(G, \mathcal{M})$ satisfy $|p_i - q_i| \le \tau$ for all i, then provided that $\tau \le 1/k$*

$$|M_i(\vec{p}) - M_i(\vec{q})| \le 2^{k+1} k\tau$$

Proof. Indeed for any $\vec{p}$

$$M_i(\vec{p}) = \sum_{\vec{x} \in \{0,1\}^k} M_i(x) \prod_{i=1}^k p^{1-x_i} (1 - p_i)_i^x,$$

therefore:

$$|M_i(\vec{p}) - M_i(\vec{q})| \le \sum_{\vec{x} \in \{0,1\}^k} |M_i(\vec{x}Z)||\alpha(\vec{x}) - \beta(\vec{x})| \le \sum_{\vec{x} \in \{0,1\}^k} |\alpha(\vec{x}) - \beta(\vec{x})|$$

where $\alpha(\vec{x}) = \prod_{i=1}^k p^{1-x_i}(1-p_i)_i^x$ and $\beta(\vec{x}) = \prod_{i=1}^k q^{1-x_i}(1-q_i)_i^x$. Since $\tau \le \frac{1}{k}$, we can immediately apply lemma 1. ■

Lemma 3 *For any Nash Equilibrium p, if $\tau \leq \frac{1}{k}$ then the nearest q in the τ-grid is $2^{k+2}k\tau$-Nash equilibrium for $(G, \mathcal{M})$*

Proof. Indeed, let r_i be the best response for player i to $\vec{q}$. Then we have:

$$M_i(\vec{q}[i : r_i]) <= M_i(\vec{p}[i : r_i]) + 2^{k+1}\tau \leq M_i(\vec{p}) + 2^{k+1}\tau \leq M_i(\vec{q}) + 2^{k+2}\tau$$

where first and last transitions follow from lemma 2 and second from the fact that p is Nash Equilibria. ■

This lemma ensures that if $\tau \leq \frac{\varepsilon}{k2^{k+2}}$, then we will have approximate NE on τ-grid.

Now we describe modifications which should be made to the aglorithm:

- The algorithm now takes additional input ε. We pick $\tau = \frac{\varepsilon}{k2^{k+2}}$
- For every node table $t_U(u, v)$ now contains only τ-discretized strategies.
- Instead of chosing the best strategy we choose ε-best strategy (e.g. which is worse the best strategy by no more than ε)

Lemma 3 assures that there is always exist a ε-nash equilibria, and by picking ε-best response we will always find it.

For running time analysis we note, that each table $t_U(u, v)$ now has size $(\frac{1}{\tau})^2$ and we have to check k whether each element is zero or one. For that we check whether there is an assignment $\{u_i\}$ for the children of U such that u is the best response to the parent playing v and $t_{U_i}(u_i, u)$ is 1. Each check could be done in time $O(2^k)$, so the whole computation could be done in $O(2^k \times \frac{1}{\tau^{k+2}})$. For fixed k, this time is polynomial in $1/\varepsilon$, n and the length of the input. However in general case this time is only quasipolynomial in the length of the input.

5 Exact algorithm

It can be shown that areas where $T_U(u, v)$ is equal to one, can be repsented as a union of at most exponentially (in number of players) many rectangles on the unit square. Moreover it is possible to compute values of $t_U(u, v)$, given $t_{U_i}(u_i, v)$, where U_i are all children of U. This gives all missing steps of the algorithm. We are not going to any further detail here, and refer the reader to [1].

References

[1] M. Keans, M. Littman, S. Singh "Graphical models for Game Theory," *Proc. International Joint Conference in Artificial Intelligence*, 1999.