CS295 Introduction to Algorithmic Game Theory

Instructor: Ioannis Panageas Scribed by: Shion Fukuzawa, Stelios Stavroulakis

Lecture 10. Other Equilibrium Notions.

1 Introduction

Up to this point we have explored the notion of Nash equilibrium and techniques to approximate it in specific types of games. As seen in the last 2 lectures, computing Nash equilibrium is computationally hard. This hardness result leads to asking if we can relax the notion of Nash equilibrium so that the problem is computationally tractable. This lecture will explore several degrees of relaxation of Nash equilibrium, starting with what is called correlated equilibria. Before diving into the definition, we introduce the following example.

2 Correlated Equilibrium

Example 2.1 ***Correlated equilibrium*** *Consider the following game of chicken between two players. In this game, a player wants to go through with the dare only if the other person is going to chicken-out.*

	***C**hicken-out*	***D**are*
***C**hicken-out*	*0,0*	*-2, 1*
***D**are*	*1, -2*	*-10, -10*

Suppose that there is a third party who ***recommends*** *the set of moves (C, D), (D, C), (C, C) with probability $\frac{1}{3}$ each. This set of moves is made known to the players. The third party then makes a recommendation by choosing one of the pairs of moves, then assigning each player the corresponding move from the pair without telling them what is suggested for the other player. Since the set of moves is told to the players, they can make reasonable assumptions about what the other player will play.*

Suppose that the row player is assigned to play D. Given the set of recommendations, the only possible assignment of the column player is C. Assuming the column player plays the assigned move, the row player will not deviate from their assigned strategy, getting an expected payoff of 1.

Now suppose that the row player is assigned to play C. Then the column player will play either C or D with probability $\frac{1}{2}$ under the set of recommendations. If the row player plays their assigned strategy of C, they get an expected payoff of $0 \cdot \frac{1}{2} + (-2) \cdot \frac{1}{2} = -1$, whereas if they switched their strategy to D, they would get an expected payoff of $1 \cdot \frac{1}{2} + (-10) \cdot \frac{1}{2} = -4.5$. So even in this situation, the row player will not deviate from their assigned strategy.

Since the game is symmetric, the exact same can be said about the column player. Since neither player will want to deviate from their assigned strategy given the set of recommended strategies, we call the set of moves (C, D), (D, C), (C, C) with probability $\frac{1}{3}$ each a ***correlated equilibrium****.*

Before introducing into the formal definition, recall the following definitions from previous lectures.

Definition 2.1 *A game is specified by*

- *set of n players $[n] = \{1, \ldots, n\}$*
- *For each player i a set of strategies/actions S_i.*
- *set of strategy profiles $S = S_1 \times \cdots \times S_n$.*
- *Each agent i has a utility $u_i : S \to [-1, 1]$ denoting the payoff of i.*

Definition 2.2 *(**Correlated Equilibrium**). Correlated equilibrium is a distribution χ over S such that for all agents i and strategies b, b' of i*

$$\mathbb{E}_{s\sim\chi}[u_i(b, s_{-i})|s_i = b] \geq \mathbb{E}_{s\sim\chi}[u_i(b', s_{-i})|s_i = b].$$

Similarly for all agents i and swapping functions $f : S_i \to S_i$,

$$\mathbb{E}_{s\sim\chi}[u_i(s_i, s_{-i})] \geq \mathbb{E}_{s\sim\chi}[u_i(f(s_i), s_{-i})].$$

Using the example to guide our understanding of the definition, we describe correlated equilibrium as a distribution of recommended moves χ such that a player can infer what moves other players will make given a recommendation, but is still better off playing a recommended move. If we let χ be the product distribution over all possible moves, then this case of correlated equilibrium corresponds to Nash equilibrium. This insight gives us the following relation between the two notions.

Set of Nash equilibria $\subseteq$ Set of correlated equilibria

3 Coarse Correlated Equilibrium

For reasons that will be clear later, we introduce a further relaxed notion of equilibrium called coarse correlated equilibrium. As in the previous section we begin by examining an example.

Example 3.1 *Consider the following matrix representing rock paper scissors.*

	R	*P*	*S*
R	*0,0*	*-1, 1*	*1, -1*
P	*1, -1*	*0, 0*	*-1, 1*
S	*-1, 1*	*1, -1*	*0, 1*

Suppose that the actions (R, P), (R, S), (P, R), (P, S), (S, R), (S, P) are chosen with probability $\frac{1}{6}$ each. Suppose that the row player plays R. Then column player will respond with either P or S with equal probability under the distribution. Now suppose the column player deviates from the distribution and decides to to play P with higher probability. Then they will incur more loss when row plays S, since row will play S and R with equal probability.

Now consider another situations where the column player is instructed to play P. Then this player knows that the other player is playing either R or S, and thus gets an average payoff of 0. With this knowledge, the column player could change strategy to R and improve their payoff to $\frac{1}{2}$ compared to the payoff of 0 when playing the recommended action. In this case, the column player can exploit knowledge of action recommendations to improve their payoff.

Notice that this differs from correlated equlibria, since a deviation from a recommended strategy led to a better payoff.

Formally, coarse correlated equilibrium is defined as the following.

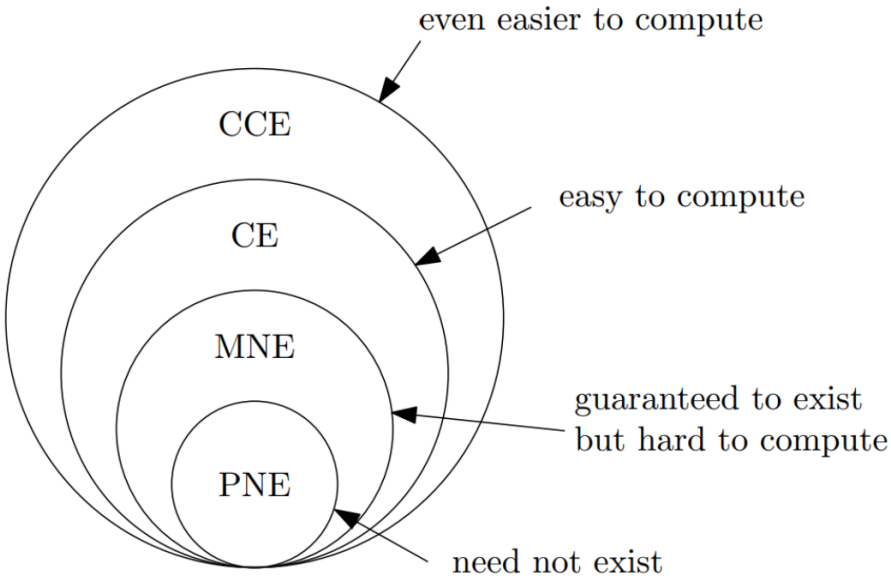
Definition 3.1 ***(Coarse Correlated Equilibrium)***. *Coase correlated equilibrium is a distribution χ over S such that for all agents i and strategies b of i*

$$\mathbb{E}_{s\sim\chi}[u_i(s)] \geq \mathbb{E}_{s\sim\chi}[u_i(b, s_{-i})].$$

The difference between coarse correlated and correlated equilibria is that we can choose a "smart" swap function that "knows" the distribution χ. By this insight we get another relation between the equilibria.

Set of correlated equilibria $\subseteq$ Set of coarse correlated equilibria.

We can summarize the relation between the notions of equilibria that we have seen so far in the following diagram.



4 Computing Coarse Correlated Equilibria (CCE)

In this section we will show a methodology used to compute coarse correlated equilibria and it will be based on no-regret learning dynamics. The reader can reference lecture 4 for further analysis of no-regret learning. Before going into detail, it is necessary to describe online learning in games which will be used to compute CCE.

Definition 4.1 *(**Online Learning in Games**). Assuming a time window T and $t \in [T]$, each player (i) chooses an action at time t from its simplex, $x_i^{(t)} \in \Delta_i$ and (after all agents have chosen their actions) experiences a payoff $u_i(x^{(t)})$ based on everyone's action. Each agent's objective is to minimize the regret they experience. Regret for each player (i) is defined as:*

$$\frac{1}{T}\left[\max_{a \in S_i} \sum_{t=1}^{T} u_i(a, x_{-i}^{(t)}) - \sum_{t=1}^{T} u_i(x^{(t)})\right]$$

The regret is calculated for each player and computes the average payoff experienced minus the best possible fixed strategy. If, while $T \to \infty$, Regret $\to 0$, the algorithm is called no-regret. Note that the comparison between average payoff and best fixed strategy is done given that the other agents didn't change their strategy while the calculation of the best fixed strategy $\frac{1}{T}\sum_{t=1}^{T} u_i(x^{(t)})$ took place.

From the family/class of all algorithms that guarantee that Regret $\to 0$ as $T \to \infty$ we choose the online gradient descent as our optimization technique.

Definition 4.2 *(**Online Gradient Descent**). Let $l_t : X \to \mathbb{R}$ be a family of convex functions, differentiable and L-Lipschitz in some compact and convex set X with diameter D. Online GD is defined as:*

Algorithm 1 Online GD

1: Initialize $x = x_0$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad y_t = x_t - a_t \nabla u_t(x_t)$
4: $\quad x_{t+1} = \Pi_\chi(y_t)$
5: **end for**
6: Regret $= \frac{1}{T}\left[\max_{a \in S_i} \sum_{t=1}^{T} u_i(a, x_{-i}^{(t)}) - \sum_{t=1}^{T} u_i(x^{(t)})\right]$

For our setting let x_0 be a strategy in the simplex Δ_i. At every iteration in the window we follow a gradient descent step. Due to the fact that X is a compact domain, we project the next point into the simplex by using the projection operator. The differentiating factor from classical gradient descent is that the function l_t changes at every time-step. Here, the function l_t represents the negative utility $-u_i(x^{(t)})$ at time t. By appropriately choosing a vanishing step size $a = \frac{D}{L\sqrt{t}}$ it holds that:

$$\left(\frac{1}{T}\sum_{t=1}^{T} u_t(x_t) - \min_x \sum_{t=1}^{T} u_t(x)\right) \leq \frac{3}{2}\frac{LD}{(T')}$$

Both *Multiplicative Weights Update* and *Online Gradient Descent* give the same guarantees i.e. ϵ error in $\Theta(\frac{1}{\epsilon^2})$ iterations. Alternatively, the regret after T iterates is $O(\frac{1}{\sqrt{T}})$ which approaches 0 as $T \to \infty$.

Proof:

Let $x^\star$ be the argmin of $\sum l_t(x)$,

$$\begin{aligned}
l_t(x_t) - l_t(x^\star) &\leq \nabla l_t(x_t)^T(x_t - x^\star), \text{ value dominates first order approximation from convexity} \\
&= \frac{1}{a_t}(x_t - y_t)(x_t - x^\star), \text{ from the definition of online GD} \\
&= \frac{1}{2a_t}(||x_t - x^\star||_2^2 + ||x_t - y_t||_2^2 - ||y_t - x^\star||_2^2), \text{ from the law of Cosines} \\
&= \frac{1}{2a_t}(||x_t - x^\star||_2^2 + ||y_t - x^\star||_2^2) + \frac{a_t}{2}||\nabla u_t(x_t)||_2^2, \text{ from the definition of } y_t \\
&= \frac{1}{2a_t}(||x_t - x^\star||_2^2 + ||y_t - x^\star||_2^2) + \frac{L^2 a_t}{2}, \text{ since } l_t \text{ is L-Lipschitz} \\
&= \frac{1}{2a_t}(||x_t - x^\star||_2^2 + ||x_{t+1} - x^\star||_2^2) + \frac{L^2 a_t}{2}, \text{ by taking the projection}
\end{aligned} \tag{1}$$

Let us now consider the sum of the differences $u_t(x_t) - u_t(x^\star)$ for all $t \in T$

$$\begin{aligned}
\sum_{t=1}^{T}(u_t(x_t) - u_t(x^\star)) &\leq \sum_{t=1}^{T} ||x_t - x^\star||_2^2 \left(\frac{1}{2a_t} - \frac{1}{2a_{t-1}}\right) + \frac{L^2}{2}\sum_{t=1}^{T} a_t \\
&\leq \frac{D^2}{2}\sum_{t=1}^{T}(\frac{1}{a_t} - \frac{1}{a_{t-1}}) + \frac{L^2}{2}\sum_{t=1}^{T} a_t \\
&\leq \frac{D^2}{2a_t} + \frac{L^2}{2}\sum_{t=1}^{T} a_t \\
&\leq \frac{LD}{2}\sqrt{T} + 2\sqrt{T}\frac{LD}{2}, \text{ since } \sum \frac{1}{\sqrt{t}} \leq 2\sqrt{T} \text{ and } a_t = \frac{D}{\sqrt{t}L}
\end{aligned} \tag{2}$$

■

4.1 A fast algorithm to approximate CCE

Suppose each agent i uses no-regret dynamics with with learning rate $a_t = \frac{\sqrt{n}}{\sqrt{t}}$ since we assume Lipschitz constant is 1 and diameter D is the product of simplices which is equal to $\sqrt{n}$ by using

Cauchy-Schwarz inequality. Recall $x^{(t)}$ is the mixed strategy profile at iterate t.

Also, suppose that σ^t is the product distribution on the strategy profiles S induced by $x^{(t)}$. We now define σ as the uniform distribution over all product distributions $\sigma^{(t)}$ which is not necessarily a product distribution. Generally, a convex combination of product distributions is not necessarily a product distribution. If that was the case, it could be easy to find Nash Equilibira. By defining σ and $\sigma^{(t)}$ we have:

$$\begin{aligned} \mathbb{E}_{s\sim\sigma} &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{s\sim\sigma^t}[u_i(s)] \\ \min_{b\in S_i}\mathbb{E}_{s\sim\sigma}\left[u_i(b,s_{-i})\right] &= \min_{b\in S_i}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{s\sim\sigma^t}\left[u_i(b,s_{-i})\right] \end{aligned} \tag{3}$$

The above is a result of the law of total expectation since the expectation of u_i drawm from σ is the sum of expectations of u_i drawn from σ^t on average. Next, we combine equation (3) with the fact that *Online Gradient Descent* is a *No-Regret* algorithm.

It is true that the expectation of player's i utility (drawn from σ) versus the best hindsight strategy for player i called $b^\star$ is bounded by the no regret algorithm and it doesn't grow larger than $\frac{3}{2}\frac{LD}{(T')}$

$$\min_{b\in S_i}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{s\sim\sigma^t}\left[u_i(b,s_{-i})\right] - \mathbb{E}_{s\sim\sigma}[u_i(s)] \leq \frac{3}{2}\frac{\sqrt{n}}{\sqrt{T}}$$

Here the diameter in this product of simplices is $\sqrt{n}$ and the Lipschitz constant is 1. By carefully choosing $T = \frac{9n}{4\epsilon^2}$ the above bound becomes ϵ and we end up with an ϵ-*approximate Coarse Correlated Equilibrium.*

The above can be summarized elegantly with the following punchline. **If every player chooses a *No-Regret* algorithm then the result is an ϵ-approximate coarse-correlated equilibrium**. Above, we considered online gradient descent which is a no-regret algorithm. Note that the distribution is not a *product distribution.* If it were a product distribution then we would have ϵ-approximate Nash Equilibrium and not ϵ-approximate CCE.
It is finally important to mention that if we alternatively used *Multiplicative Weights Update* the iterates required would also be inversely proportional to the inverse fraction of ϵ^2 but logarithmic with respect to n:

$$\mathsf{MWUA} \rightarrow O\left(\frac{ln(n)}{\epsilon^2}\right)$$

which is faster than the methodology mentioned above. If accuracy is of our concern, they demonstrate similar performance.

4.2 CE vs CCE

As a final note, now that we showed that computing CCE is easy, it is important to know what is in general easy to compute, based on the hierarchy diagram of equilibria. *Correlated Equilibria* is also easy to compute provided we use a different kind of regret called *swap regret*. The framework of *no-regret* can be used to create a *No Swap Regret* algorithm and efficiently find approximate CE.

References

[1] Names of authors. *Name of article*. Publisher.