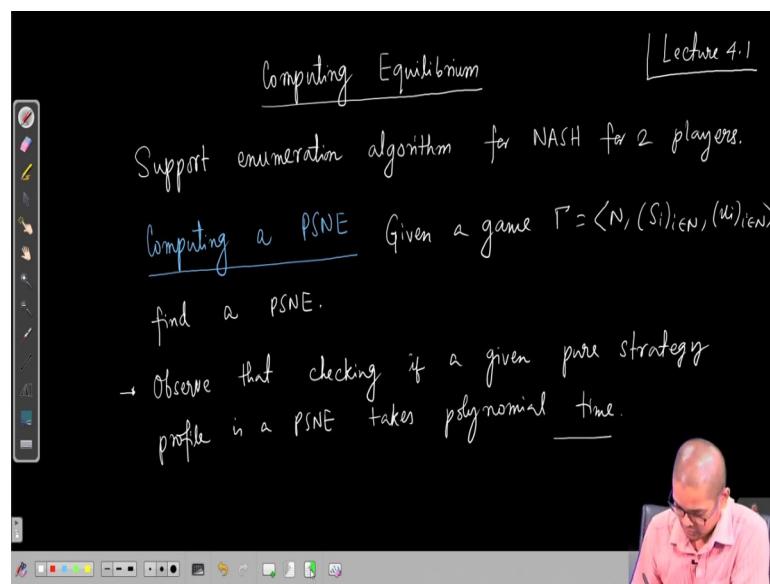


Algorithmic Game Theory
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 16
Succinct game

Welcome. In the last class, we started the computational problem of finding equilibrium. So, we will continue this for few lectures now.

(Refer Slide Time: 00:35)

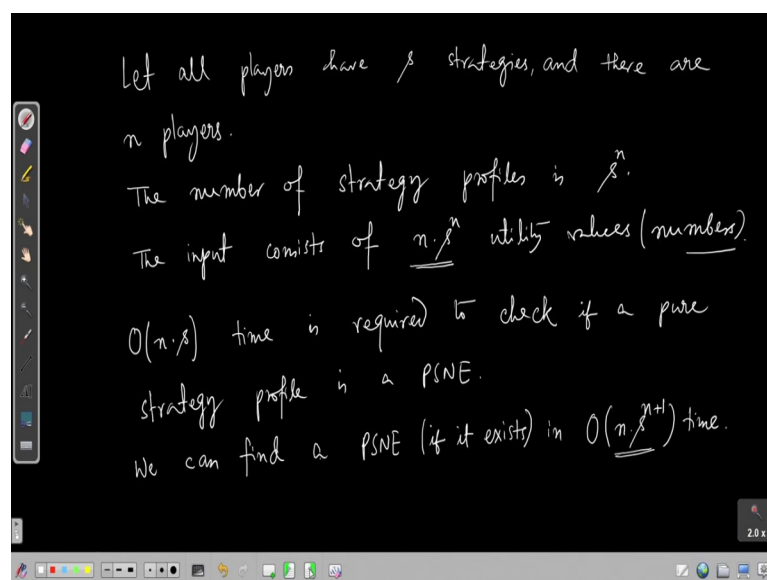


Computing equilibrium and in the last class we have seen a support enumeration algorithm for NASH problem for 2 players. So, what was the NASH problem? Given a game in; given a game in mixed strategies given a game in normal form, we need to find a mixed strategy NASH equilibrium. But what, but why mixed strategy NASH equilibrium? Before that what about computing pure strategy NASH equilibrium.

So, let us spend some time on the problem of finding pure strategy NASH equilibrium today. So, or not finding the paned computing, computing a PSNE. What do you mean by computing? Given a game in normal form, we need to find a pure strategy NASH equilibrium. Given a game $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$, given a game in normal form find a PSNE; Pure Strategy NASH Equilibrium.

Now, I say that this problem is trivial, let us understand. So, why it is very easy? So, first observe that checking if a given pure strategy profile is a PSNE takes polynomial time, why? So, why polynomial? So, how? So, let us first write down what is input length.

(Refer Slide Time: 04:21)



So, for simplicity let us make some assumptions. So, let us have let all the players, let all players have s strategies and there are n players. So, how many; so, ok. So, what is the number of strategy profiles? So, the number of strategy profiles is s^n . Now, for each strategy profile I need to tell what is the utility in the strategy profile of each player. So, the input consists of n times s to the power n utility values which are basically numbers, real numbers or rational numbers. So, for computation we can assume that these are rational numbers only.

So, when we say that something is polynomial, we mean polynomial in the input size; polynomial in n and s^n . Now, given a strategy profile how can we check whether the a given pure strategy profile is a PSNE or not? We go over all players and look at unilateral deviations. Now, there are n players and for each player there are s many unilateral deviations. So, in n times s in this time is required to check if a pure strategy profile is a PSNE.

So, the simple algorithm of simply iterating over all strategy profiles, all s^n strategy profiles and spend $O(n \cdot s)$ time to check whether that strategy profile is a PSNE or not.

The total time required is $n s^n$. So, we can find of PSNE of course, if it exists in $O(n s^{n+1})$ time which is polynomial in input size which is $n s^n$.

So, that is why we say that a PSNE can be computed in polynomial time in general. But, you know there are some special kind of games, there are some important classes of games where it is not necessary to list down explicitly all these $n s^n$ utility values. There are other succinct ways to represent the game and if it is possible to represent the game in in space much less than n times s to the power n plus 1, then it is not the case that this algorithm is polynomial time runs in polynomial time. So, let us see a few such important classes of games.

(Refer Slide Time: 09:18)

Important classes of Succinct Games

1. Graphical games: We have a directed graph G on the set N of players. The utility of a player i depends only on the players who have a directed edge to i , including i . If the in-degree of G is at most d , then (n/s^{d+1}) numbers are needed to represent the game.

The diagram shows two nodes, i and j . Node i has d incoming edges from other nodes, and node j has one incoming edge from i .

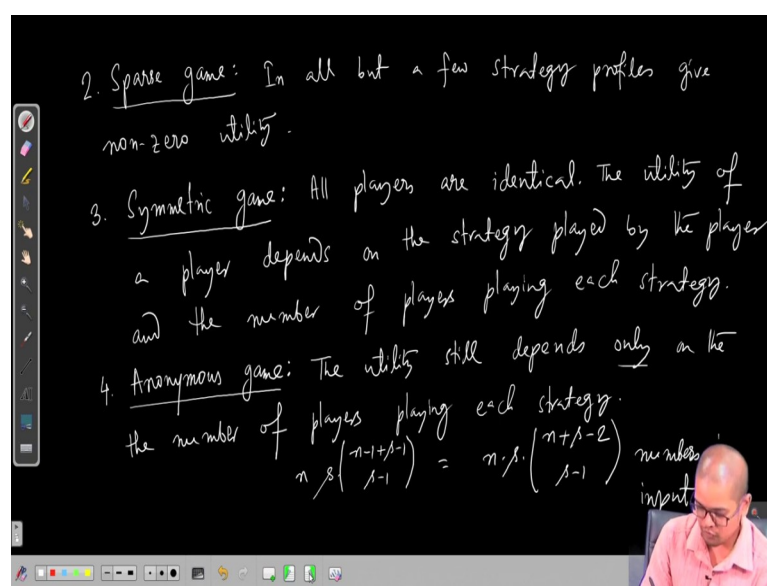
So, those games are called succinct games, important classes of succinct games. Our first example is a graphical games, its a very well studied class of games. So, we have a directed graph G on the set N of players ok. And, what does this graph signifies? Is that the utility of a player i depends only on the players who have a directed edge to i ok.

And of course, the utility of player i depends on i itself including i . So, if here is the player i and here is player j ; and if I have a directed edge from i to j ; that means, player i 's utility depends on the action or the strategy played by the player j . So, if so, what is the input size now? If the in degree of G is at most d , in degree means that you for every vertex how many incoming edges are there.

Now, for every vertex there are d many incoming edges; so, we just need to vary. So, if there are d many incoming edges to i say then we just the; so, the utility of player i depends on the strategy played by this d in neighbours and the and the strategy played by i . So, the strategy profile that we for which you need to write down the utility value of player i is s^d for those n neighbours and of course, there are s many strategies for player i .

So, s to the power d plus 1, this is the number of strategy profiles for which I need to write down the utility of player i and this this I need to do for all the for each player and there are n players. So, it is n times this. Then, this many numbers are needed to represent the game. So, if d is small, you see this is a significant savings. If t is constant say this is much less than n to the $n s^{n+1}$.

(Refer Slide Time: 14:59)



Our second class of games is what is called sparse games. So, in this type of game utilities are 0 or always 0, except for few strategy profiles. So, in all, but a few strategy profiles give non-zero utility. So, if the game is like that then instead of listing down all those 0's, you can simply skip them. And, for all those strategy profiles which has some non-zero utility, we can list them down and for them we can write down what is the utility of each player in those strategy profiles. So, that is sparse game.

Our third important class of game is symmetric games. So, here all players are identical. Hence, what do you mean by that? That means, the utility of a player depends on

because all players identical their strategy set is also same. And, the utility depends on the strategy played by the player of course, and the number of players playing each strategy.

So, suppose there are 10 strategies and I am a player, my utility depends only on say how many players play strategy 1 and how many players play strategy 2, how many players play strategy 3 and so on. In particular, if I take two strategy profiles, but the number of players who play strategy 1 is same in both the strategy profiles and number of players who play strategy 2 is same for both the profiles and so on.

And that means, for every strategy the number of players played the number of players who play that strategy is same in both the strategy profiles, then for me the my utility will be same. And, my utility depends only on the strategy I played. And, all the players who play that for example, suppose I have played strategy 5 and suppose there are 10 players who have played strategy number 5. Then, all those 10 players who have played strategy number 5 will receive the same utility; that is what we mean by players are identical.

The names of the players does not matter. The utility matrix are symmetric and it only depends on how many how many play how many how many players play a particular strategy. Typically, voting situations are very natural where this symmetric games are used. Symmetric games capture the voting setting various voting settings very aptly.

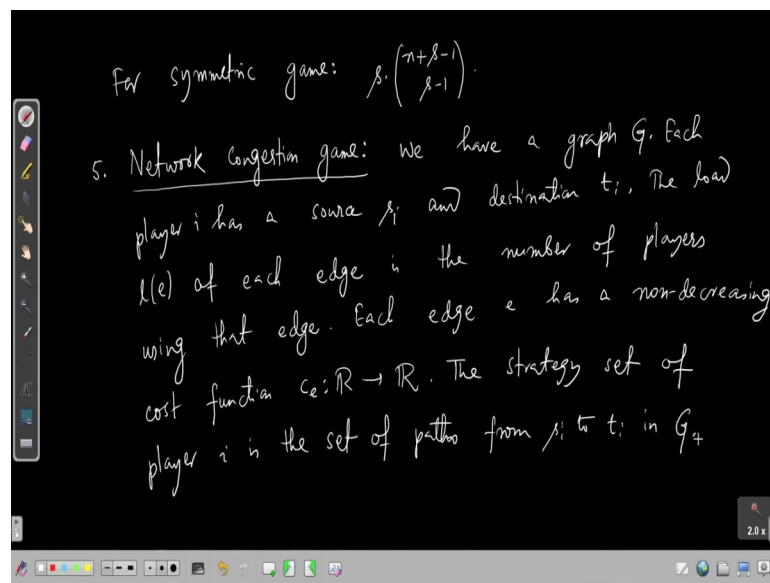
For example, to decide the winner it does not matter who voted for which candidate. All it matters is that how many votes, how many voters have voted candidate 1, how many voters have voted candidate 2 and so on. The our next example is anonymous games. It is a generalization of symmetric game and the utility still depends on the. So, the utility still depends only on the number of players playing each strategy.

But, the difference from symmetric game is that you know the utility the players who have played the same strategy, they can receive different utilities that can happen for anonymous games. So, what is the number of numbers we need to write to represent anonymous games? So, focus on one player, fix one player. It has s many strategies to play. And, what is the other players how many ways other players can play?

So, and we will count we will if two strategy profiles give rise to the give rise to each strategy being played by same number of players, we will count them as 1. So, that way the number of remaining players is $n-1$ and there are for each of them there are s strategies. So, $^{n-1}C_{s-1}$.

So, these are the number of numbers I need to write for each player and there are n players. So, these $ns^{n+s-2}C_{s-1}$, numbers in input. What was what is the number of numbers I need to write for symmetric game?

(Refer Slide Time: 22:57)

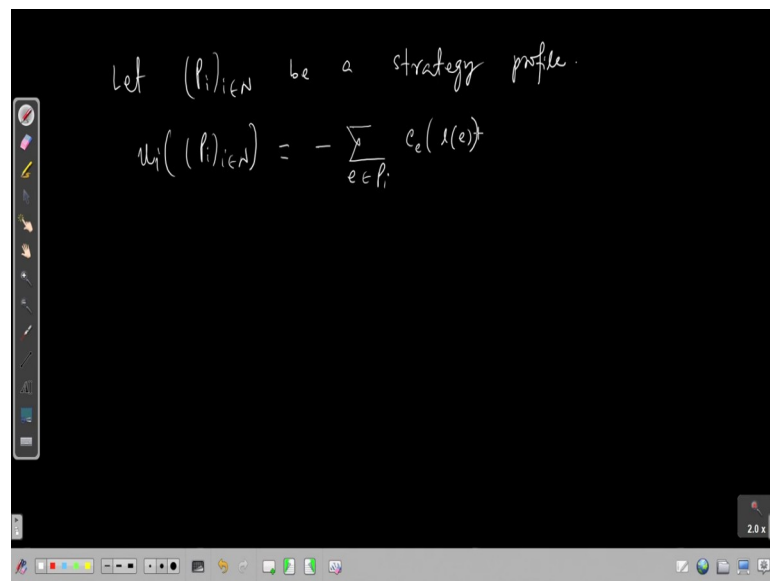


For symmetric game, how many strategy profiles in this case? n players and each of them as s strategy profile, s many strategies $^{n+s-1}C_{s-1}$. And, for each strategy profile I need to tell for each strategy what is the utility, because all players who play same strategy will receive the same utility; so, s times this ok. So, this is the number of numbers I need for symmetric games.

Our next example is network congestion games, network congestion game. So, here we have a graph G and each player has a source and destination, has a source s say s_i ; each player i has a source s_i and destination t_i ok. And, for the load of each edge, the load l_e of each edge is the number of players using that edge ok. And, for each edge there is a congestion function or cost function.

Each edge e has a non-decreasing cost function c_e which depends on load which is the function from say set of natural numbers load to or it is better to write it as a set of real numbers to real numbers, its non-decreasing. If the load increases, the cost increases. It is often useful to encode delay to represent delay in this thing this way. And, the strategy set of each player, strategy set of player i is the set of paths from s_i to t_i in G ok. And, how will you compute the utility?

(Refer Slide Time: 27:27)



$$\text{let } (p_i)_{i \in N} \text{ be a strategy profile.}$$

$$u_i((p_i)_{i \in N}) = - \sum_{e \in p_i} c_e(l(e))$$

So, let $(P_i)_{i \in N}$ be a strategy profile. So, the utility of player say i in this strategy profile $(P_i)_{i \in N}$ is because we are working with cost we can write minus summation of edges in this path c_e of l of e . Look at the load in this edge and from the load you compute what is its cost. And, you sum over the cost of the paths along sum of the sum over the cost of those edges in along the path and that is the cost. And if you want to write utility, you take minus of it. So, this is what is called network congestion game.

So, in next class we will see some more important class of succinct games ok.