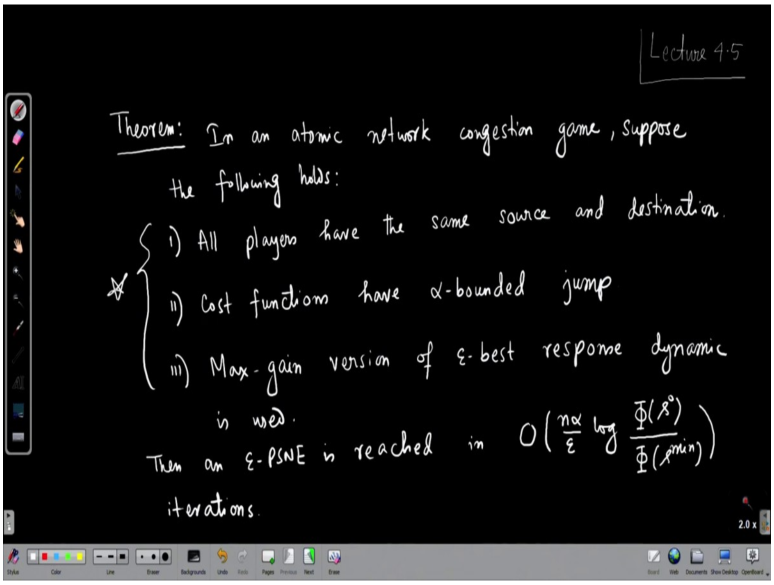
Algorithmic Game Theory
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Computing ε-PSNE for Network Congestion Games

Welcome, so from last few classes we were studying potential games and it is an important class of games which has an which has a pure strategy NASH Equilibrium PSNE. And the reason it is called potential game because these games admit what is called a potential function and we have seen various important examples of potential games and most notably is Network Congestion Game and hence in the network congestion game there exist a PSNE.

Not only that in the last class we proved a very important theorem let me briefly recall the theorem.

(Refer Slide Time: 01:06)

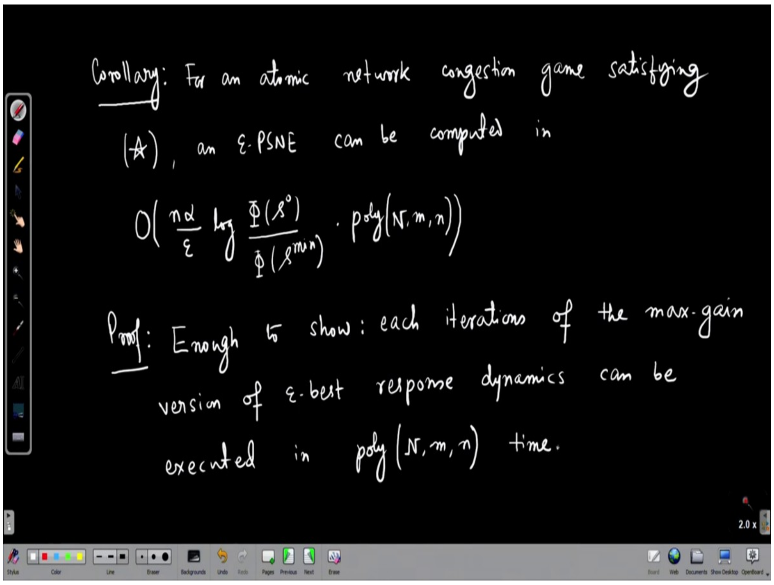


Which basically says that in a network in an atomic network congestion game, suppose the following holds some conditions; so, what are the conditions? 1st is all players have the same source and destination, 2nd cost functions satisfy bounded jump property, 3rd max gain version of epsilon best response dynamic is used. Then an ϵPSNE is reached in

$$O\left(\frac{n\alpha}{\epsilon}\log\left(\frac{\phi\left(s^0\right)}{\phi\left(s^{min}\right)}\right)\right).$$

Where s^0 is the initial strategy profile by $\phi\left(s^{min}\right)$ and ϵ PSNE is reached. In this many iterations of the max gain version of ϵ best response dynamic.

(Refer Slide Time: 04:59)

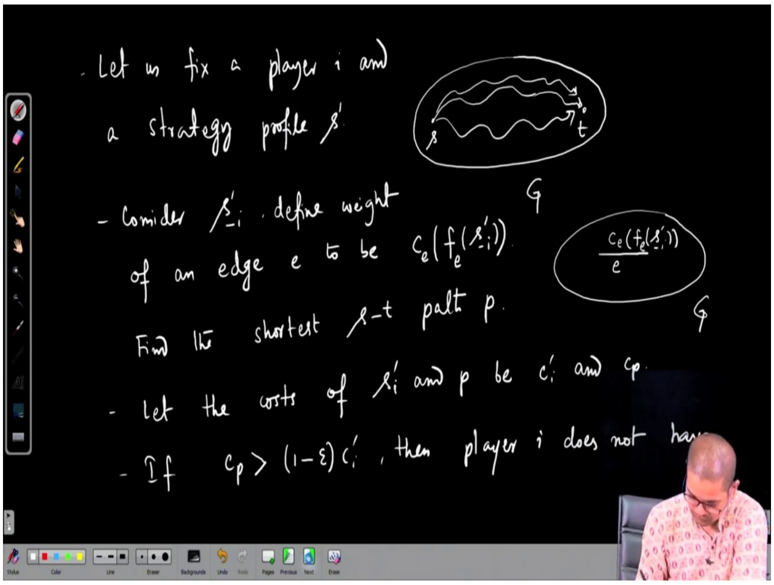


So, from this what immediately follows is that for network congestion game with those properties means, for an atomic network congestion game satisfying the previous set of conditions. Let us call this set of conditions to be star satisfying star those conditions, an epsilon PSNE can be computed in $O\left(\frac{n\alpha}{\epsilon}\log\left(\frac{\phi\left(s^0\right)}{\phi\left(s^{min}\right)}\right)\right)$ these are the number of iterations.

And every iteration can be executed in polynomial time polynomial in N, m is the number of edges, n is the number of vertices let me write capital N, because there are 2 n s small n is the number of players and capital N is the number of vertices in the graph ok proof. So, all we need to show is that each iteration of the max gain version of epsilon best response dynamic can be executed in polynomial time. So, to show or enough to show there should be small n also there enough to show.

What each iterations of the max gain version of epsilon best response dynamics can be executed in polynomial? In capital N number of vertices in the graph m number of edges and small n number of players in this time. How that is what we will explain now?

(Refer Slide Time: 09:17)



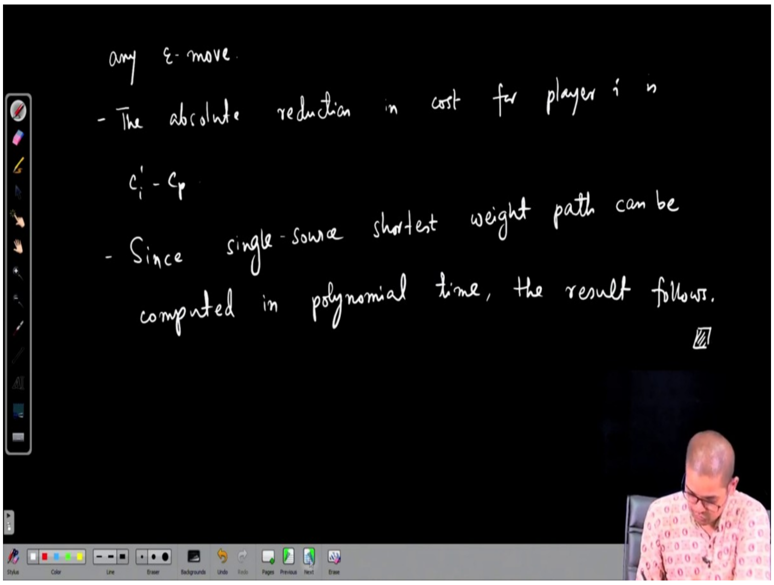
So, suppose this is the graph G and so let us fix a player i and a strategy profile s. What is the strategy profile? So, there is a source and there is a destination, so let us call it s prime because I am using source s and t destination and each player is picking a path from source to s to t, this paths can overlap also ok.

Now for player i we want to find what is the best available path for player i, the best available deviation. So, what we do is that we find from s or consider s prime minus i look at the strategy profile of all players except i and that that is is also a collection of paths and each edge has a cost depending on how many players are using that edge.

And in that graph so consider another graph same graph G with same set of vertices and same set of edges, but the cost of this edge e is $c_e(f_e)$ look at the flow in the strategy profile s_{-i} that is the cost. And in this graph you find the shortest s to t path define weight of an edge e to be look at the cost function of e look at the flow in this edge e in the strategy profile s'_{-i} ok. Now find the shortest s-t path in this graph G and that is the best possible deviation for player i and see how much absolute reduction of cost it incurs.

So, shortest s to t path let us call it P. So, let the costs of $s_i^{'}$, $s_i^{'}$ is the path that player i was choosing in the strategy profile $s^{'}$ at the cost of $s_i^{'}$ and this newly discovered shortest path P be $c_i^{'}$ and c_p. Now if c_p is more than $(1-\epsilon)c_i^{'}$, then player i does not have any ϵ move.

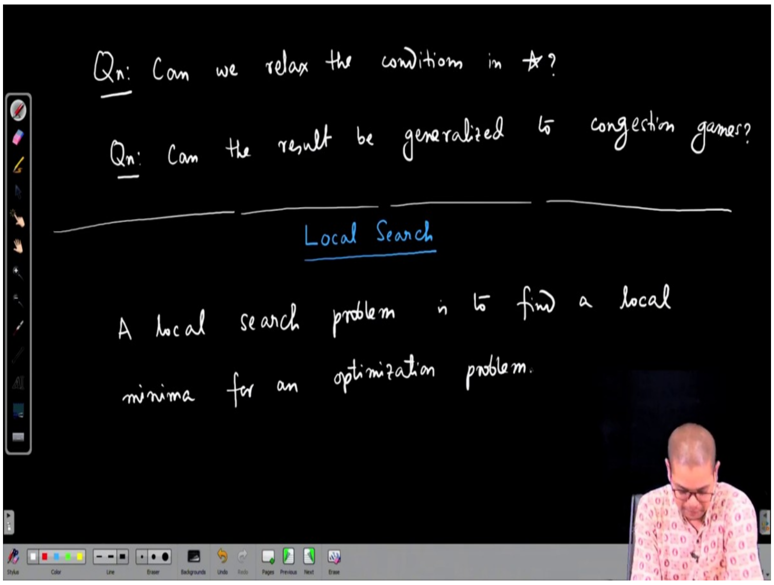
(Refer Slide Time: 14:10)



And the absolute reduction in cost for player I, if it allowed to deviate from $s_i^{'}$ to p is $c_i^{'}-c_p$. So, for every player we have found what is the best possible highest possible absolute reduction of cost for that player i and once this data is there we can check which players has an epsilon move; among those players which has an epsilon move I can pick that player and that particular path which gives the highest absolute reduction in the cost of player i and hence that is all what we need to do in the i th in the in the iteration.

So, since shortest path shortest weight path since single source shortest weight path can be computed in polynomial time the result follows. So, we need polynomially many iterations and each iteration can be executed in polynomial time, that is hence an ϵ PSNE for a network congestion game under those conditions can be computed in polynomial time by simply running the max gain version of ϵ best response dynamics. Now the question is a natural question is so this concludes the proof of the corollary.

(Refer Slide Time: 17:04)



So, first question is can we relax the conditions in star? So for example, does the result still follow if all players does not have same source and destination, it seems that it does not follow and we do not hope to have a polynomial time algorithm for finding an epsilon best response dynamic when source and destinations are not all the same. But how will you prove it? So, that we will see next, but let me point out another question can the result be generalized to congestion games.

Recall congestion games are a generalization of network congestion game and the question is for a congestion game also can we find an ϵ best response can we find a ϵ PSNE in polynomial time. And for both the questions the answer is no with some qualification and that machinery we will develop now.

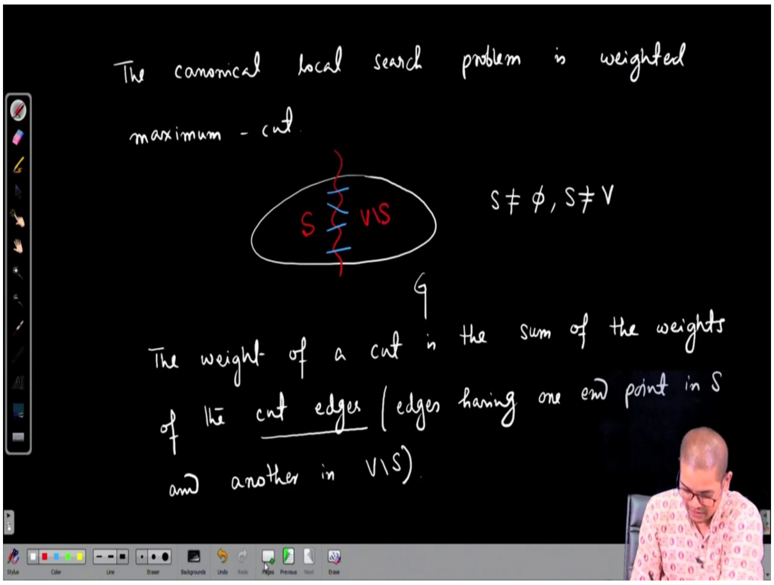
So, for typical computer science problems there is a machinery of np completeness and np hardness which allows us to make claims like some problem is NP hard and NP complete; thereby implying that we do not hope to have a fast or efficient algorithm for these problems. Can we prove that this problem is NP complete?

So, in general and we will see that there is there is some subtle issues, but the 1st road block is that these are not these are not yes no answer. Typically the machinery of np completeness works only for decision problems whose outputs are either yes or no. But if you if you sort of try to pose this or change rephrase these questions in a decision version then the then the problem becomes trivial.

For example, the 2nd question or the 1st question what is the 1st question? Can we find the epsilon PSNE for a network congestion game without the condition that all source and destinations being same. Now if you if i want to ask a decision version of it a decision version could be that does there exist an epsilon PSNE for a network congestion game, where source and destination of all the players need not be same. But you see that the answer is yes the answer is always yes, the decision version of these problem are trivial.

So, that is why typical machinery of NP hardness that theory does not work and we need to build up a new fresh new theory. Towards that we define we start today a topic of local search. So, what is local search? A local search problem is to find a local minima or maxima for an optimization problem. So, what is the canonical problem of local search, for NP the canonical problem is set.

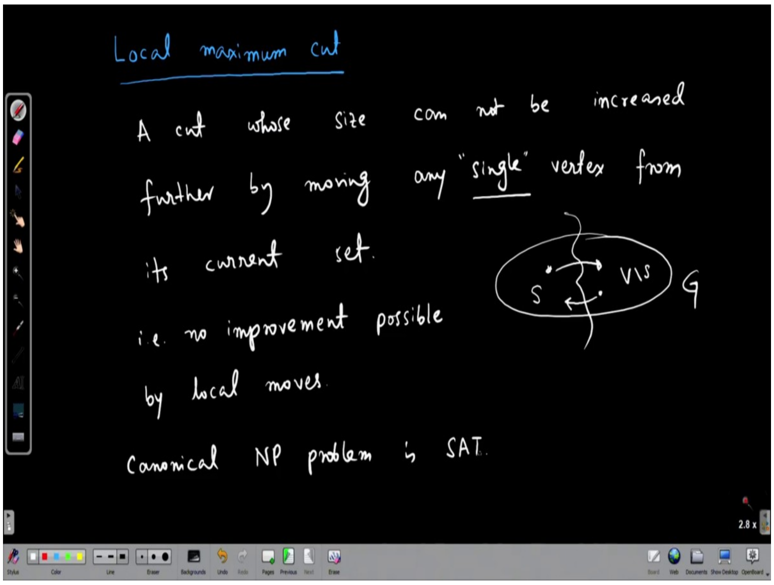
(Refer Slide Time: 22:23)



Similarly, for local search the canonical local search problem is weighted max cut. What is this problem I am given a graph G and I want to find a cut is $(S, V \setminus S)$. What is the cut? It is a; it is a partition of the set of vertices into 2 non-empty sets. So, S is not empty set and $S \neq V$. And what is the weight of this cut? There are various edges which cross this cut, we say an edge crosses a cut if its endpoints are in both parts. So, the weight of a cut is the sum of the weights of the cut edges.

So, what are which are cut edges? Edges having 1 end point in S and another in $V \setminus S$. And what is the local search version of weighted max cut? The weight the usual weighted max cut problem is that given a weighted graph find a cut of size maximum a maximum size cut. But what is the local version of max cut problem what is local max cut?

(Refer Slide Time: 25:47)



So, let me use some other colour local maximum cut it is a cut whose size cannot be increased further by moving any single vertex from it is current set. What do you mean by that? So, this is the graph G and suppose I am looking at this cut. Now we can take 1 vertex at a time and ask if I move this vertex from this set S to say $V \setminus S$ in the new cut that I get what is the size of this new cut does it increase or decrease or remain same.

Similarly for a vertex in V minus S also for every vertex we try to we can try to change it is set. Now if it happens that for all vertex it is not possible to increase the size of the size of the cut by moving any single vertex. The single is very important any single vertex, then that is called a Local maximum cut. Please note that it is still possible that a local maximum cut can its size can be increased further by moving more than 1 vertices at a time. So, it may be possible that if I simultaneously move two vertices a and b and change their change their position with respect to the cut, then I get a get a cut of strictly larger size.

But by moving only one single vertex it is not possible then such a cut is called a local maximum cut. So, that is and those sort of moves are called local moves, we are only making we are only changing 1 vertex and also making small change. So, that is no improvement possible by local moves ok. So, this is our canonical local search problem and so this is like we had a canonical np problem is like SAT, it is like this local max cut in the class local search is same as the role that SAT plays in the class NP.

So, in the next class we will formally define what is local search problem, because the problems are not decision problems we need formal definition and then we will see why do you what do we mean by that we do not expect to have a polynomial time algorithm for finding an epsilon PSNE for congestion game and so on ok.

Thank you.