Ubinet, Distributed Optimization and Games 2015-2016 February 7, 2016

Lecture 7

Lecturer: Giovanni Neglia *Scribe: Jimmy Auk, Hakob Melkonyan*

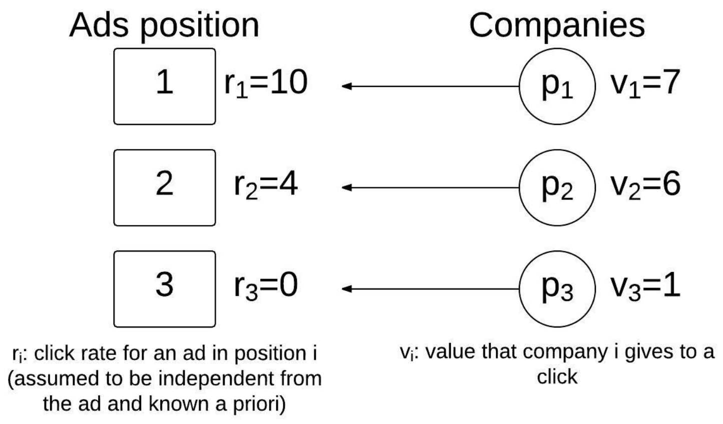
NOTE: The content of these notes has not been formally reviewed by the lecturer. It is recommended that they are read critically.

In the previous lesson we have studied VCG mechanism for auctions to ensure a truth-telling mechanism. This time, the emphasis will be on the GSP mechanism used by Google Inc. and game design principles.

1 GSP auction mechanism

Google Inc. has adopted the GSP (Generalized Second Price Auction) mechanism for getting bids for its ad places. Compared to VCG (Vickrey-Clarke-Groves) auction, in GSP every bidder is paying the amount equal to the next lower bid from its own bid. Note that this is the natural way to generalize to the second-price one-item auction. But one may ask the following question: "Why use GSP instead of VCG?" As we are going to see through the following simple example, GSP may offer the possibility for the auctioneer to earn more money than VCG.

Let us suppose that there are 3 items to be sold to 3 bidders, and each item has its click rate r_1, r_2, r_3 respectively. We also denote each player's values as v_1, v_2, v_3 respectively.


Ads position
Companies
1
r_1=10
p_1
v_1=7
2
r_2=4
p_2
v_2=6
3
r_3=0
p_3
v_3=1
r_i: click rate for an ad in position i (assumed to be independent from the ad and known a priori)
v_i: value that company i gives to a click

With the click rates and values shown in the diagram above, if player 1 is honest and bids the actual price that he/she values the item, then will get position 1 and his/her pay-off will be: 10(7-6)=10. But if instead he/she bids 5, then will get position 2 and the pay-off will be 4(7-1)=24. This simple example shows that in GSP truth-telling may not be the best strategy for individual players.

In terms of Game Theory, there is a Nash-Equilibrium (NE) if none of the bidders has any incentive to change his/her strategy. In the example above there are 2 NEs as follows: when p_1 bids 5, p_2 bids 3, p_3 bids 2, and when p_1 bids 3, p_2 bids 5, p_3 bids 1. In order to make sure that a given set of bids is NE, we need to check for each player, if changing his/her bid will make any increase in his/her pay-off. If there is no increase in pay-off for a given player, then we can say that the given bid is the Best-Response strategy to other bidders' strategies for that particular player. It is possible to prove that there always exist a socially optimal NE, in the sense that it maximizes total advertisers' evaluation.

Now, let's compare the total revenue for the auctioneer in each case of NE. For the first NE values described above, the total revenue will be 48, while for the second NE it will be 34. This example shows that the revenue equivalence principle does not hold for auctions with multiple goods. Moreover, if we calculate the seller revenue under VCG, that would be 44. We find then with GSP the seller revenue can be both higher or smaller than under VCG. The reasons behind Google's choice for GSP remains unclear.

In addition, there are several general issues like the real value of the click, which remains uncertain and changes dynamically; high bids by low-quality advertisers; and complex queries for which no direct bid has been specified. To cope with those issues, there are several techniques to estimate the real value of the click and to extrapolate bids from bids for simpler queries, but those remain to a large extent under secret from public.

Before terminating this part on auctions, we want to introduce two definitions.

The science that studies how to design a game in order to modify the users' payoffs so that the final output of the game is a desired one is called *Mechanism Design* or also *reverse game theory.* In Mechanism Design economic incentives are usually introduced to the purpose to modify the intrinsic (and usually unknown) player's utility function. If the incentives are such that truth-telling becomes a dominant strategy, then the designed mechanism is called *incentive compatible.*

2 Revisiting the Network Utility Maximization problem

In the Network Utility Maximization problem, we have seen that the decomposition approach leads to a socially optimal allocation when users are price takers (i.e. when they ignore how prices are calculated) and then they solve the following optimization problem:

$$\max_{w_r} U_r(\tfrac{w_r}{\lambda_r}) - w_r$$

But once the price anticipators appear in the game, the social optimality is in general lost, because they solve the following problem instead:

$$\max_{w_r} U_r(\tfrac{w_r}{\lambda_r(w_r)}) - w_r$$

A way to this issue once and for all is to design a corresponding VCG payment mechanism. The basic idea is that each user r is asked for his/her utility function. A priori he/she may lie and declare a utility function $\hat{U}_r$ different from his/her real one. The SYSTEM problem is then solved considering the functions $\hat{U}_r$ for each $r \in \mathcal{R}$, but each user, say it r, is required a payment corresponding to the social loss for all the other players due to the presence of r. Let $U_S(B)$ be the social utility calculated over a given set B of declared utility functions. User r will be charged:

$$Payment_r = U_S(\hat{U}_v, v \in R - \{\text{r}\}) - (U_S(\hat{U}_v, v \in R) - \hat{U}_r)$$

Here, the first term of the right side is what the others would get without player r, and the second term is what altogether get except what r gets. It is possible to prove that this scheme is incentive compatible, i.e. r will tell the truth instead of lying by providing $\hat{U}_r = U_r$.

But in order to implement this VCG-approachwe have the following issues on our way:

1) How to transmit user's utility function.
2) Network needs to solve $|R| + 1$ optimization problems.
3) We don't know how to do it in a distributed fashion.

Because of the difficulty to design incentive compatible mechanims, researchers have also taken a different approach, trying to quantify the performance loss due to the presence of rational selfish users. For a given system designed, as soon as rational selfish users can decide among multiple actions, a game arise. In general there will be multiple NEs with different corresponding social utilities. How do the social utility at one of this NE compare with the optimal value (e.g. achievable through centralized optimization)? There will be some Loss of Efficiency (LoE) which we can quantify as follows:

$$\text{LoE at a NE } = \frac{\sum_r U_r(x_r^*)}{\sum_r U_r(x_r^{NE})} \geqslant 1$$

where $\mathbf{x}^*$ is the social optimal allocation and $\mathbf{x}^{NE}$ is the allocation at the NE. The LoE for the NE with the lowest efficiency value is called the *Price of Anarchy* (PoA) of the games, and sometimes it may be calculated even without being able to calculate all the other NEs. There is also another quantity called *Price of Stability*, which is the LoE for the NE with the highest total outcome.

$$PoA = \max_{\text{over all NEs}} \frac{\sum_r U_r(x_r^*)}{\sum_r U_r(s_r^{NE})} \geqslant 1$$

$$PoS = \min_{\text{over all NEs}} \frac{\sum_r U_r(x_r^*)}{\sum_r U_r(s_r^{NE})} \geqslant 1$$

Example 1. *Calculating Loss Of Efficiency (LOE)*
Given

$$\textit{Optimal utility } U_S^* = 1$$
$$NE_1 \textit{such that } U_S^1 = 0.99$$
$$NE_2 \textit{such that } U_S^2 = 0.5$$

we can calculate

$$PoA = \frac{1}{0.5} = 2$$
$$PoS = \frac{1}{0.99} = 1.01$$

For the network utility maximization problem, it is easy to proe that for a network with 1 single shared link : PoA $= \frac{4}{3}$. It is also possible to prove that for general networks : PoA $\leqslant 4\sqrt{2} - 0.5$

3 Engineering Systems with the Game Theory

Game Theory is of "descriptive" nature and is not intended for solving distributed problems but it still may be applied in some cases to design systems of agents to accomplish a common task. We can think that this approach is close to Mechanism Design, but in mechanism design users have their private utility function and we introduce incentives to align it with our purpose. Here, instead, we consider a situation where we can design from scratch the whole system, players' utilities included.

While this approach is working and giving satisfactory results, one may ask the following question: "Why Game Theory should be used to engineer systems while there are other general approaches?" There are 3 main answers to this question:
1) It is intrinsically distributed.
2) There are lot of results in Game Theory about Partial Information (PI), i.e. about players not knowing the other players' utilities or even their own.
3) In Game Theory there's a lot about learning, i.e. about how to reach a given NE.

3.1 How to design a Game

The process of designing a game is basically designing the payoff matrix, so we may end-up with one or more NE.

Let's consider such a game where two drones should make a decision to attack robots. If there are 2 robots attacking drones, and one of them is closer to the first drone and the other one is closer to the second drone, then each drone would prefer to attack a robot together, because if they will not cooperate they will have a bad outcome. It's not so important if they both go first left or right, but make the same decision together. This problem is well known as "Battle of sexes", where a couple needs to decide where to go on the weekend (for example either to movie or to opera), and each of them prefer to go anywhere together rather than alone.

So we design a game by "placing" NEs and decide how we want them to converge to the preferred NE, e.g. by asynchronous best response dynamics. There are following possibilities:

- Loops may happen in the payoff matrix preventing the asynchronous best response dynamics to converge to any NE.
- It may converge to NE slowly.
- It may only converge to a pure strategy NE.
- It may require too much information.

3.2 Potential games

Let us adopt the following notation:

N – the number of players in the game
A_i – set of actions of player i (a finite set here)

a_i – action of player i
$\underline{a}$ – vector of the actions, that we can also denote as (a_i, a_{-i}) to highlight player i's action
The game has a potential function φ if

$$\exists \varphi : A_1 \times A_2 \times ... * A_N \to \mathbb{R}$$

that has the following property:

$$U_i(a_i', a_{-i}) - U_i(a_i'', a_{-i}) = \varphi(a_i', a_{-i}) - \varphi(a_{-i}'', a_{-i}) \qquad \forall a_{-i}, \forall i$$

In such games, that are called Potential games (because of the potential function), the asynchronous best response dynamic (BRD) always converges to a NE. If fact, imagine the player i first selects its BR, then U_i will increase and also φ will increase. Similarly, if player j selects then his/her BR, then U_j will increasing, and with it U_i may increase or decrease, but φ will increase again. Because the set of possible outcomes is finite, φ has at least a local maximum and after a finite number of iterations, the BRD will reach a local maximum. From now on, no BR will cause a change in the actions because φ cannot grow anymore. The dynamics has then reached a NE.

The example of designing a routing game as a potential game is missing.

References

[1] Srinivas Shakkottai and R. Srikant, "Network Optimization and Control." 2008.

[2] Jason R. Marden and Jeff S. Shamma, "Game Theory and Distributed Control." 2012.

[3] David Easley and Jon Kleinberg, "Networks, Crowds, and Markets: Reasoning about a Highly Connected World." 2010.