Algorithmic Game Theory and Applications

Lecture 16: a brief glimpse beyond

Kousha Etessami

School of informatics

warning, again

- The subjects we will briefly glimpse at today are part of a fast emerging field at the intersection of game theory, CS, economics and e-commerce, and "internetology".

  Numerous advanced courses are being taught on these subjects around the world, and there are conferences and workshops dedicated to them. See, e.g., the on-line lecture notes for C. Papadimitriou's graduate course at Berkeley 2001-2003; and the proceedings of the ACM conference on Electronic Commerce.

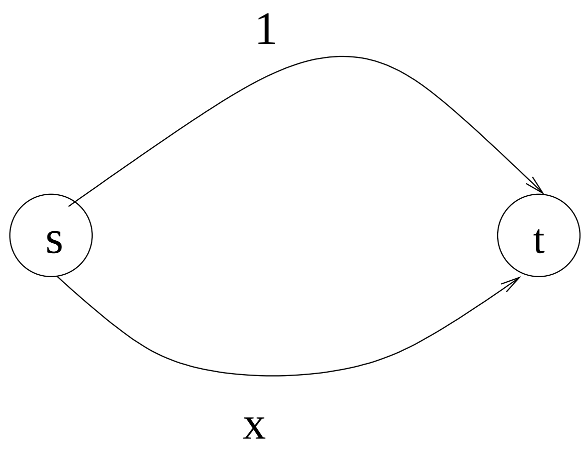
- We won't even scratch the surface (of the surface) of these subjects, so please do explore them further if they interest you.

- As a starting point, please read today's handout, the brief but very engaging survey:
  "Algorithms, Games, and the Internet",
  [Christos Papadimitriou, STOC'2001].

games and the internet

- Basic idea: "The internet is a huge experiment in interaction between agents (both human and automated)".
- Such interactions can profitably be viewed from a game theoretic viewpoint: agents trying to maximize their own payoffs, etc.
- How do we set up the rules of these games to harness "socially optimal" results?
- These vague notions can best be illustrated by examples.

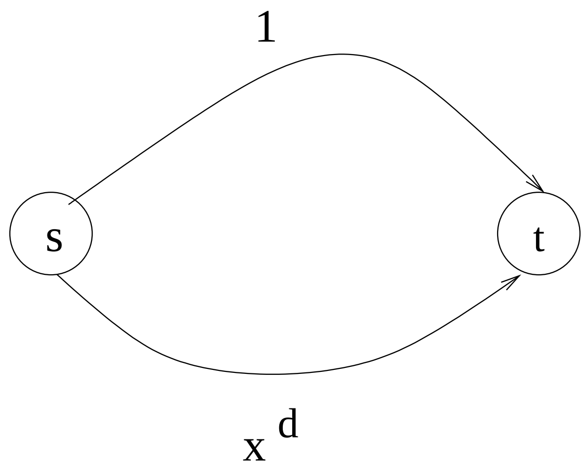
a flow network game



(from [Roughgarden-Tardos'00])

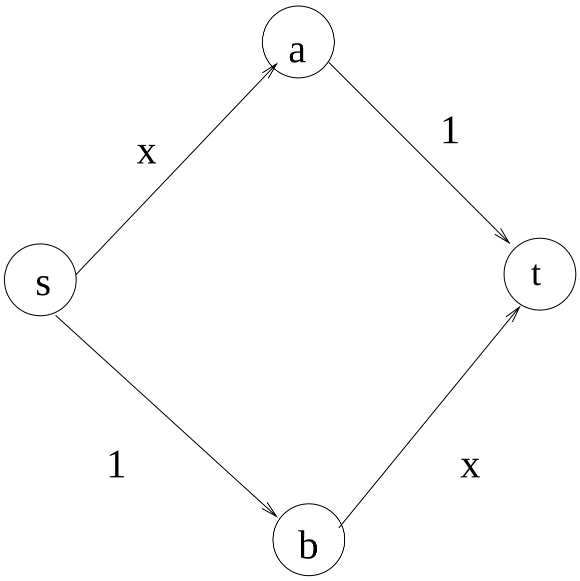
- A large number n of customers in the network want to go from s to t.
- Each can either take the edge labeled 1, with "latency" 1 (delay of crossing edge), or edge labeled with latency x. Here x represents the "congestion", given by the ratio of the number of customers that are using that edge divided by the total n.
- Assume n is very large, (basically, $n \rightarrow \infty$).
- What is the Nash Equilibrium?
  (NEs in such a setting can be shown to be essentially unique [Beckmann, et. al. '56].)
- What is a "globally optimal" customer routing strategy profile that minimizes average delay? What is the globally optimal average delay?

a modified game



- What is the NE, and what is the average delay it induces?
- What is the globally optimal average delay?

a different network

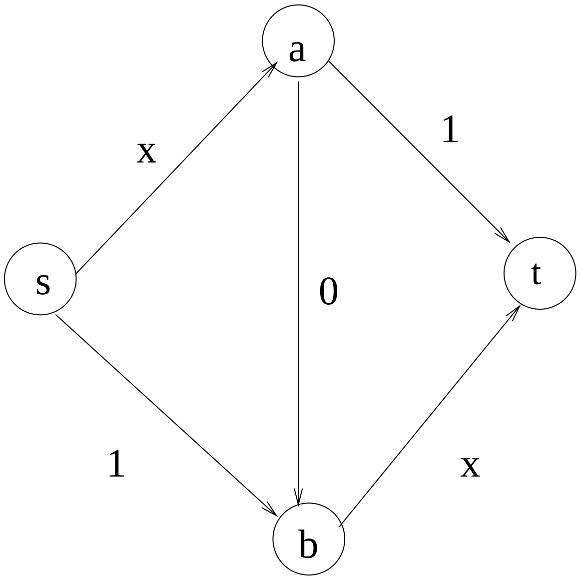


- What is the NE, and what is its average delay?

- What is a globally optimal strategy profile and optimal average delay?

- What if an ambitious “network service provider” wanted to build additional “superfast” lines?

Braess's paradox



- What is the NE and its average delay?
- What is the globally optimal average delay?

social welfare and the price of anarchy

Recall that in a strategic game Γ, we may have different measures of the "social welfare" *welfare*(x) under a particular profile of (mixed) strategies $x \in X$. For example, "utilitarian" social welfare is *welfare*$(x) := \sum_{i=1}^{n} U_i(x)$. For a game Γ, let $NE(\Gamma)$ be the set of NE's of Γ.

For our next definition suppose *welfare*$(x) > 0$ for all $x \in X$. (In many games we could enforce this by, e.g., "shifting" all payoffs by an additive factor.)

A version of "<u>the price of anarchy</u>" can be defined as: ([Koutsoupias-Papadimitriou'98,Papadimitriou'01])

$$PA(\Gamma) := \frac{\max_{x \in X} \textit{welfare}(x)}{\min_{x \in \mathrm{NE}(\Gamma)} \textit{welfare}(x)}$$

Thus, the "price of anarchy" is the <u>ratio</u> of best "global" outcome to the the worst NE outcome. Note: this ratio is ≥ 1 and larger means "worse". (Perhaps "price" is not the best word for this variant.)

It would be comforting to establish that in various situations the "price of anarchy" isn't too high.

price of anarchy in the flow network game

- For flow f let *welfare*$(f) := 1/$(average s-t-delay).

- In Braess's paradox, the price of anarchy is $4/3$: by playing the NE the average delay is 2, but playing half-and-half on the upper and lower route, the average delay is $3/2$ (and that's optimal).

- We have seen that the price of anarchy in network games can be arbitrarily high, when x^d is an edge label.

- Remarkably, [Roughgarden-Tardos'00] showed that in a more general flow network setting (where there can be multiple source-destination pairs (s_j, t_j)), as long as "congestions" labeling edges are linear functions of x, the worst-case price of anarchy is $4/3$.

- In other words, for linear latencies, Braess's paradox is the worst-case scenario.

auctions as games

Auctions have been studied game theoretically for a long time. Consider one such formulation:

- Each of n bidders is a player.
- Each player i has a "valuation" $v_i \in \mathbb{R}$ for the item being bid on.
- The payoff to player i, if he/she wins the auction at price pr_i is
$$u_i(outcome) := v_i - pr_i$$
- All other players get payoff 0.
- Under the following constraints, the auctioneer is free to set up the rules of the auction: given bids $(b_1, \ldots, b_n)$ one of the highest bidders must win, and at a price $\leq \max_i b_i$.
- Question: What rules should the auctioneer employ, so that for each player i, revealing the "true worth" v_i is a dominant strategy (i.e., $b_i = v_i$ is a dominant strategy)?

vickery auctions and mechanism design

- In a Vickery auction, a.k.a., *second-price, sealed bid* auction, the highest bidder wins but the price paid by the highest bidder is the second highest bid price.

- Convince yourself: bidding your true "valuation" v_i is a dominant strategy in this game.

- This is the starting point of a vast sub-discipline of game theory called "Mechanism Design", where the goal is to design a game where selfish player will behave in a desired way (e.g., a "socially optimal" way).

- Many algorithmic issues impinge on mechanism design. See, e.g., [Nisan-Ronen'99,Nisan'99-00, . . .].

a formal setting for mechanism design

Formally, the input to a mechanism design problem can be specified by giving three things:

- An **environment**, $\mathcal{E}$, in which the game designer operates.
- A **choice function**, f, which describes the designer's preferred outcomes in the given environment.
- A **solution concept**, *Sol*, which describes the kinds of strategy profiles of games (such as their Nash Equilibria) which will be considered "solutions" in which the desired choice function should be implemented.

We describe each of these three inputs separately, using auctions as a running example to illustrate each concept.

Once this is done, we will be able to state the mechanism design problem formally.

environments

An **environment**, $\mathcal{E} = (N, C, \Theta, \mathcal{M})$, consists of:

- A set $N = \{1, \ldots, n\}$ of players.
- A set C of "outcomes".
  auction example: C could be a set of pairs (i, pr), meaning that player i wins and pays price *pr*.
- A cartesian product $\Theta = \Theta_1 \times \Theta_2 \times \ldots \times \Theta_n$, where each set Θ_i is a set of possible "utility functions" for player i. Each $u_i \in \Theta_i$, is a function $u_i : C \mapsto \mathbb{R}$, that maps outcomes to payoffs for player i.
  auction example: u_i could be given by
  $$u_i((i', \mathit{pr})) = \begin{cases} 0 & \text{if } i \neq i' \\ v_i - \mathit{pr} & \text{if } i = i' \end{cases}$$
- A set $\mathcal{M}$ of "mechanisms". Each $M \in \mathcal{M}$ has form $M = (S_1, \ldots, S_n, g)$, where S_i is a set of strategies for player i, and $g : S_1 \times \ldots \times S_n \mapsto C$, is a function from strategy profiles to "outcomes".
  auction example: the strategy sets S_i give the possible bids for each player, and g gives a function that maps a profile of bids to an "outcome" of who wins and at what price. Such outcome functions can be constrained (using $\mathcal{M}$), e.g., to only allow top bidders to win, and to ask for at most the maximum bid price to be paid.

choice functions

Given an environment $\mathcal{E}$, a **choice function**

$$f : \Theta \mapsto 2^C$$

for the designer, specifies what set of outcomes the designer would prefer if it knew exactly what utility (payoff) function each player has.
auction example: the "truth revealing" choice function the designer would prefer could be given by: $f((u_1, \ldots, u_n)) = \{(i, v_i) \mid v_i = \max_{i \in N} v_i\}$.

Note that a mechanism $M = (S_1, \ldots, S_n, g) \in \mathcal{M}$ together with the tuple $u \in \Theta$, gives a complete description of a strategic game $\Gamma_{M,u}$, where the payoff function for player i can now be described as a function of strategy profiles: by abuse of notation, for the game $\Gamma_{M,u}$ we can write $u_i(g(s_1, \ldots, s_n))$ as simply $u_i(s_1, \ldots, s_n)$.

solution concepts

We lastly need the notion of a "solution concept" which specifies what kind of solutions of a game we are trying to implement the choice function in. Such solutions might be Nash Equilibria, or more strong conditions such as profiles of dominant strategies.

Given an environment $\mathcal{E}$, a **solution concept** is given by a mapping *Sol*, whose domain is $\mathcal{M} \times \Theta$, and such that for each mechanism $M = (S_1, \ldots, S_n, g)$ and each tuple $u = (u_1, \ldots, u_n)$ of utility functions, *Sol*$((S_1, \ldots, S_n, g), u) \in 2^S$, where $S = S_1 \times \ldots \times S_n$. auction example: in the auction example we may wish to implement the choice in dominant strategies. In this setting, we ask that *Sol*(M, u) be the set of strategy combinations $(s_1, \ldots, s_n)$ such that each s_i is a dominant strategy for player i in $\Gamma_{M,u}$.

finally: the mechanism design problem statement

The mechanism design problem can now be stated formally:

> Given environment $\mathcal{E} = (N, C, \Theta, \mathcal{M})$, a choice function f, and a solution concept *Sol*, find $M = (S_1, \ldots, S_n, g) \in \mathcal{M}$ such that for all tuples $u \in \Theta$ of utility functions, *Sol*(M, u) is a non-empty set and for all $s \in$ *Sol*(M, u), $g(s) \in f(u)$.

If such a mechanism M exists, we say M *Sol*-**implements** choice function f in environment $\mathcal{E}$. We then also say f is *Sol*-**implementable** in $\mathcal{E}$.

This was a rather long-winded definition. There is room to argue whether some of this formalization adequately captures "real life" situations of mechanism design. Nevertheless, these formalizations have proved useful to auction design and the mechanism design community. See books [Mas-Colell-Whinston-Green'95,Osborne-Rubinstein'94].

One fact that may at first seem surprising is the "revelation principle".

the revelation principle

Define *Dom* to be the solution concept where $s = (s_1, \ldots, s_n) \in S$ is in *Dom*(M, u) if and only if for every player i, s_i is a dominant strategy in $\Gamma_{M,u}$. Such a profile s is called a dominant strategy equilibrium (DSE) of $\Gamma_{M,u}$.

In environment $\mathcal{E} = (N, C, \Theta, \mathcal{M})$, a mechanism M is called a **direct revelation mechanism** if $M = (\Theta_1, \ldots, \Theta_n, g)$. In other words, the strategies for each player amount to "announcing" what its utility function is. Players may in general "lie", but.....

Proposition (*The Revelation Principle*) Suppose f is *Dom*-implementable in $\mathcal{E} = (N, C, \Theta, \mathcal{M})$. Then in environment $(N, C, \Theta, \mathcal{M}')$, where $\mathcal{M}'$ consists of all direct revelation mechanisms, there is some $M' = (\Theta_1, \ldots, \Theta_n, g') \in \mathcal{M}'$ such that for all $u \in \Theta$, $u \in$ *Dom*(M', u) and $g'(u) \in f(u)$.

Thus, each player revealing their true utility function is a DSE in $\Gamma_{M',u}$. We then say M' **truthfully** ***Dom*-implements** f. Intuitively, the revelation principle says if f can be *Dom*-implemented then there is a revelation mechanism where each player "might as well" reveal their true utility.

combinatorial auctions

As mentioned, many algorithmic issues impinge on mechanism design. We unfortunately haven't time to get into these. See, e.g., [Nisan-Ronen'99,Nisan'99-00]. The last thing we will look at is:

- Consider an auction where instead of one item, a set of k items, K, is up for auction at the same time and each bidder i may only want to bid on some subset $D_i \subseteq K$, for a price b_i.

  Question: Given bids $(D_1, b_1), (D_2, b_2), \ldots, (D_n, b_n)$ what is the most profitable allocation for the auctioneer?

  Obviously NP-hard. (Max Clique even when all bids b_i are the same.)

- If bidders have more complicated criteria for which subsets they want to bid on, how should they express this? I.e., with what "bidding language"?

- Again, there is a large literature on combinatorial auctions and its connections to mechanism design. See, e.g., [Nisan'00,Parkes'00, . . .].

THE END

(hope you enjoyed the course)