Algorithmic Game Theory and Applications

Lecture 19: Auctions and Mechanism Design III: Matching Markets, unit-demand auctions, and VCG; and a Formal Framework of Mechanism Design

Kousha Etessami

Matching markets: multi-item unit-demand auctions

(Now start thinking of, e.g., Google's Sponsored Search Auctions.)

- A set B of n bidders (Advertisers);
  A set Q, of k items (Advertisement slots on your Google page).
- Each bidder wants to buy at most one item (at most one ad on a web page),
  i.e., each bidder has unit demand.
- Each bidder $i \in B$ has a valuation, $v_{i,j} \geq 0$, for each item j.
- Each item j has a reserve price, $r_j \geq 0$, below which it won't be sold.
- A (feasible) price vector is any vector $p \geq r$.
- At prices p, the demand set of bidder i is
  $D_i(p) := \{j \in \arg\max_{j' \in Q}(v_{i,j'} - p_{j'}) \mid v_{i,j} - p_j \geq 0\}$.
- The payoff (utility), u_i, to a bidder i who gets item j and pays p_j is $u_i = v_{i,j} - p_j$. (And $u_i = 0$ if bidder i gets nothing.)

Market equilibrium in matching markets

- A partial matching, μ, is a partial one-to-one function from B to Q.
- A price vector p, together with a partial matching μ, is called a Market price equilibrium, (p, μ), if:
  - For all $i \in B$, if $\mu(i) = j$, then $j \in D_i(p)$.
  - If $i \in B$ is unmatched in μ, then for any $j \in D_i(p)$, $v_{i,j} - p_j = 0$.
  - For any item $j \notin \mu(B)$, $p_j = r_j$.

Theorem ([Shapley-Shubik,1971])

1. *A market equilibrium always exists.*
2. *The set P of market equilibrium price vectors p form a lattice in $\mathbb{R}^k_{\geq 0}$.*

The least element of this price lattice has very special properties.
(For maximizing revenue, sellers would prefer the greatest element, but...)

Computing market equilibrium: an ascending price auction

[Demange-Gale-Sotomayor,1986]

1. Assume (only for simplicity) that valuations $v_{i,j}$ are all integers.
2. Set the initial price vector, p^0, to the reserve prices, $p^0 := r$.
3. At round t, when current prices are p^t, each bidder i declare their demand set, $D_i(p^t)$.
4. Does there exist an overdemanded set of items, $S \subseteq Q$? If not, STOP. By Hall's theorem, a market equilibrium at prices p^t exists (and a corresponding matching can be found maximum matching).
5. If so, find a minimal over-demanded set S, and for all $j \in S$, $p_j^{t+1} := p_j^t + 1$. Set $t := t + 1$, and go to step 3.

Theorem ([DGS'86])

- *This auction algorithm always finds a market equilibrium.*
- *Moreover, the equilibrium it finds is the least price equilibrium.*

Some algorithmic considerations

- Can we compute this auction outcome efficiently? Yes.
- We can compute (minimal) over-demanded sets in P-time.
- We don't need to increment prices by 1 in each iteration.
  (We can raise prices in the over-demanded set until the demand set of one of the respective bidders enlarges.)
- It turns out that this is essentially isomorphic to the Primal-Dual Hungarian Algorithm for finding a maximum-weight matching in a weighted bipartite graph [Kuhn'55], which runs in P-time.

More lovely facts about this lovely auction

- As we said, the DGS auction always computes the minimum price equilibrium, p^*.
- In fact, it turns out that the payments, p^*, are precisely the VCG payments!
- Therefore, the mechanism is incentive compatible, and for every bidder it is a dominant strategy to specify its true valuations.
- Moreover, this mechanism is even group strategy-proof, meaning that no strict subset of bidders who can collude have an incentive to misrepresent their true valuations.

Generalized Second Price (GSP) auctions

The GSP auction looks somewhat similar to Vickrey's second-price auction.

1. Assume there are k advertisement slots on the web page.
   We assume there is a preference total ordering on the advertisement slots $1, \ldots, k$: all bidders prefer slot 1 to 2, slot 2 to 3, etc.
   (It can be assumed (for simplicity) that associated with each slot j is a clickthrough rate, α_j, and that for $j < j'$, we have $\alpha_j \geq \alpha_{j'}$.)
2. Each advertiser i, chooses a single bid value, $b_i \in \mathbb{R}_{\geq 0}$. (This is a bid associated with a click on their ad.)
3. The advertisers are ordered (from highest to smallest) based on their bid b_i. Let $b_{(j)}$ denote the bid that is ranked j'th.
4. If advertiser i is ranked $j < k$, then it gets its ad in position j, and pays, per click, $b_{(j+1)}$, the bid by the advertiser in position $j + 1$.
5. If there are fewer bidders than slots, then the advertiser in position k pays a reserve price r (per click).

What are the incentive properties of the GSP auction?

- It is not quite VCG (but it is related).
  Given equivalent bids by all bidders, GSP payments are at least as high (or higher) than VCG.
- Truth telling is not a dominant strategy under GSP.
- But ([Edelman-Ostrovsky-Schwarz'07],[Varian'07]) if bidding is modeled as a repeated game, then it is argued bids will eventually reach an locally envy-free equilibrium. Such equilibria of the GSP auction map directly to market equilibria of a matching market.
- Thus, we again have a lattice structure to price equilibria, and there is a minimum equilibrium in which payoffs correspond precisely to VCG. (But it is not a dominant strategy of GSP, just one equilibrium.)
- If we end up at any other equilibrium, the revenue of the seller is higher than VCG. (Not bad for Google.)

Complications: budget limits

- In actual GSP ad auctions, bidders are asked to submit not just a value bid saying how much they value a clickthrough, but also a budget limit, which specifies the maximum they are willing to actually pay (which may be strictly less).
- This creates a discontinuity in the player's utility function, and in fact (in degenerate cases), market equilibria need not exist.
- However, when bids and budgets satisfy certain basic non-degeneracy conditions: (1) market equilibria do exist, (2) the DGS auction algorithm works, and finds incentive compatible prices and allocations.

A formal framework 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 (Bayes) 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 single-item auctions as a running example to illustrate each concept.

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

environments

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

- A set $N = \{1, \ldots, n\}$ of players.
- A set C of "outcomes".
  auction example: $C = \{(i, \mathrm{pr}) \in N \times \mathbb{R}_{\geq 0}\}$, where outcome (i, pr) means player i wins and pays pr.
- A cartesian product $\Theta = \Theta_1 \times \Theta_2 \times \ldots \times \Theta_n$, of types where each Θ_i is a set of possible types for player i.
- We assume that the type $\theta_i \in \Theta_i$ of player i determines its "utility function" over outcomes. So, we have $u_i : \Theta_i \times C \to \mathbb{R}$.
  auction example:
  $$u_i(v_i, (i', \mathrm{pr})) = \begin{cases} 0 & \text{if } i \neq i' \\ v_i - \mathrm{pr} & \text{if } i = i' \end{cases}$$
  Note: the "type", v_i, of player i determines its utility function, u_i.
- $p : \Theta \to [0, 1]$ is a common prior joint distribution on types of players.
- $\mathcal{M}$ is a set of "Mechanisms"... Question: What is a Mechanism?

What is a Mechanism?

- Each Mechanism, $M \in \mathcal{M}$, has the form $M = (A_1, \ldots, A_n, g)$, where A_i is a set of actions for player i, and $g : A_1 \times \ldots \times A_n \to C$, is a outcome function from strategy profiles to outcomes.
  auction example: the action sets A_i are the possible bids that each player i could bid, and g gives a function that maps a profile of bids to an "outcome" of who wins the item and at what price. Such outcome functions can be constrained by the definition of the set $\mathcal{M}$. For example, we may only allow top bidders to get the item, and we may ask for at most the maximum bid price to be paid by whoever gets the item.

Note that a mechanism $M = (A_1, \ldots, A_n, g) \in \mathcal{M}$ together with an environment $\mathcal{E}$, gives a full description of a (Bayesian) game, $\Gamma_{M,\mathcal{E}} = (N, A, \tilde{u}, \Theta, p)$, where the payoff function $\tilde{u}_i : A \times \Theta_i \to \mathbb{R}$ for player i is now given by: $\tilde{u}_i(a_1, \ldots, a_n, \theta_i) := u_i(g(a_1, \ldots, a_n), \theta_i)$.
Note: A (pure) strategy for player i in this Bayesian setting is a function $s_i : \Theta_i \to A_i$, i.e., a function from its types to its actions.

choice functions

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

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

specifies what set of outcomes the designer would prefer if it knew exactly what type (i.e., what utility (payoff) function) each player has.
single-item auction example: the choice function of the designer could be, e.g., to give the item to a person who values it most:

$$f(v_1, \ldots, v_n) = \{(i, pr) \mid i \in \arg\max_{i'} v_{i'}\}$$

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, e.g., (Bayes) Nash Equilibria, or more strong conditions such as a Dominant Strategies Equilibrium.

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 $M = (A_1, \ldots, A_n, g)$ and each tuple $\theta = (\theta_1, \ldots, \theta_n)$ of types, $Sol((A_1, \ldots, A_n, g), \theta) \in 2^S$, where $S = S_1 \times \ldots \times S_n$, and where $S_i = \{s_i : \Theta_i \to A_i\}$ is the strategy set of player i.

auction example: in the auction example we may wish to implement the choice in dominant strategies. In this setting, we ask that $Sol(M, \theta)$ be the set of strategy combinations $(s_1, \ldots, s_n)$ such that each $s_i : \Theta_i \to A_i$ is a dominant strategy for player i in the (Bayesian) game $\Gamma_{M,\mathcal{E}}$.

finally: the mechanism design problem statement

The mechanism design problem can now be stated as follows:

> *Given environment* $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$*, a choice function* f*, and a solution concept* Sol*, find a mechanism* $M = (A_1, \ldots, A_n, g) \in \mathcal{M}$ *such that for all tuples* $\theta \in \Theta$ *of types,* $\text{Sol}(M, \theta)$ *is a non-empty set, and for all* $s \in \text{Sol}(M, \theta)$*,* $g(s(\theta)) \in f(\theta)$*.*

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 $\mathcal{E}$.
of mechanism design. See books
[Mas-Colell-Whinston-Green'95,Osborne-Rubinstein'94].
Note: As already indicated by the *Gibbard-Satterthwaite Theorem*, in some environments, some social choice functions, such as truth revelation, are not (dominant-strategy)-implementable.
A result that is at first surprising, called the "Revelation Principle", says that everything that is dominant-strategy implementable can in fact be implemented as a truth revelation dominant strategy.

The Revelation Principle

Let *Dom* be the solution concept where $s = (s_1, \ldots, s_n) \in Dom(M, \theta)$ if and only if for every player i, $s_i : \theta_i \to A_i$ is a dominant strategy in $\Gamma_{M,\mathcal{E}}$. Such a profile s is called a dominant strategy equilibrium (DSE) of $\Gamma_{M,\mathcal{E}}$. Similarly, define *BNE* to be the solution concept where $s \in BNE(M, \theta)$ iff s is a Bayesian Nash Equilibrium of $\Gamma_{M,\mathcal{E}}$.
In environment $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$, a mechanism M is called a **direct revelation mechanism** if $M = (\Theta_1, \ldots, \Theta_n, g)$. I.e., players' strategies $s_i : \Theta_i \to \Theta_i$ amount to "announcing" their type. Players can lie, but.....

Theorem (*Revelation Principle*)

Suppose f is *Dom*-implemented (*BNE*-implemented, respectively) by $M = (A_1, ..., A_n, g)$ in environment $\mathcal{E} = (N, C, \Theta, u, p, \mathcal{M})$.
Then in environment $(N, C, \Theta, u, p, \mathcal{M}')$, 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 $\theta \in \Theta$, $s^* \in Dom(M', \theta)$ (respectively $s^* \in BNE(M', \theta)$), where $s_i^*(\theta_i) = \theta_i$ for all i. And, $\exists s \in Dom(M, \theta)$, ($\exists s \in BNE(M, \theta)$, respectively), such that $\forall \theta \in \Theta$, $g'(s^*(\theta)) = g(s(\theta)) \in f(\theta)$.

interpreting the revelation principle

The revelation principle (RP), for now restricted to the dominant-strategy case, says that if a mechanism M *Dom*-implements a choice function f then there is another mechanism M' which *Dom*-implements f, where each player revealing their true type is a DSE in $\Gamma_{M',\mathcal{E}}$. We then say M' **truthfully** ***Dom*-implements** f.

(An analogous version can be stated for implementation in BNE.)

Note the contrast to the Gibbard-Satterthwaite Theorem, which says that there is no non-dictatorial social choice function for player's preferences among 3 or more alternatives for which "truth revealing" is implementable by dominant strategies.

proof of the revelation principle

Proof

Let's prove it in the case of dominant strategy implementation.
Suppose that some indirect mechanism $M = (A_1, \ldots, A_n, g)$, *Dom*-implements f.
Suppose $s' = (s'_1, \ldots, s'_n)$ is a dominant strategy profile in the game $\Gamma_{M,\mathcal{E}}$, where $s'_i : \Theta_i \to A_i$.

In other words, each player i will prefer to play $s'_i(\theta_i)$ if his type is θ_i, regardless of what the other strategies s_{-i} of other players are.

Now we create a direct revelation mechanism $M' = (\Theta_1, \ldots, \Theta_n, g')$ by using a mediator.
The mediator says to each player i: "If you tell me your type, and if you say your type is θ_i then I will play strategy $s'_i(\theta_i)$ for you."
Clearly, since s'_i was dominant for M, telling the truth, θ_i, will be dominant for the new direct revelation mechanism M', and it will implement the same choice function f. □

In the Algorithmic Mechanism Design problem, we will additionally want to insist that functions like the outcome function $g : A_1 \times \ldots \times A_n \rightarrow C$ of the mechanism $M = (A_1, \ldots, A_n, g)$ that implements f should be efficiently computable.

THE END

(hope you enjoyed the course)