Algorithmic Game Theory	**Fall 2019**

Lecture 2: Introduction to Auctions

Scribes: James Hulett 9/24/19

2.1 Single-Item Model

We will start our discussion about auctions with the single-item model, where there are n bidders competing for a single item. We will model an auction in this scenario as the following three-step procedure (known as a *sealed-bid auction*):

(1) Each bidder i secretly submits their bid b_i to the auctioneer.

(2) The auctioneer decides who gets the item.

(3) The auctioneer decides what the winner pays for the item.

Playing the role of the auctioneer, we would like to design steps (2) and (3) such that the bidders are encouraged to give bids in step (1) that are in some way reasonable. To make this formal, we make the following assumption about our bidders' behavior:

Assumption 2.1. Each bidder i assigns some value v_i to the item, which is kept a secret from the other bidders. Each bidder then attempts to maximize their own utility, which we assume to be $v_i - p$ if they pay p for the item and 0 if they don't get the item.[1]

Using this as our model of bidder behavior, our goal is now to design our auction system such that each bidder is incentivized to tell us their actual valuation of the item; that is, we want to have $b_i = v_i$ for all i. In an intuitive sense, this is a good goal to have as it gives us as the auctioneer the most information with which to make the decisions in steps (2) and (3).

In addition to this assumption about how bidders act, we will also assume that step (2) is implemented in an (intuitively reasonable) way:

Assumption 2.2. The highest bidder always wins the item.

In later weeks, we'll see cases where using different rules to decide who wins can be helpful, but for now we will only focus on figuring out how much to charge the winner once they have been determined.

2.1.1 An "Awesome" Design: The Vickrey Auction

Last week, we discussed one "good" way of implementing the payment step: the second-price auction (also known as the Vickrey auction). In this auction, the highest bidder wins, but instead of paying their own bid, they pay whatever the *next highest* bid was. As we briefly discussed last week, this auction system is good for a number of reasons, outlined in the following theorem.

[1]This is known as a *quasilinear utility model*.

Theorem 2.3 (Vickrey '61). *The Vickrey auction is "awesome", meaning that*

(1) Having bidder i bid v_i is a dominant strategy (this strategy maximizes bidder i's utility regardless of what other bidders do).

(2) Any bidder following this strategy is guaranteed a non-negative utility.

(3) If all bidders follow their dominant strategy, the auction maximizes the social surplus $\sum_i v_i a_i$, subject to the constraint that exactly one a_i is 1 and the rest must be 0.

(4) The auction can be implemented efficiently.

An auction that has the first two properties is often called *dominant-strategy incentive-compatible (DSIC)*. These properties tell us, in effect, that the auction is easy for the bidders to play, and that there's really no reason for them to not join in. This is good for us as the auctioneer, as if more people join our auction, we're more likely to have bidders who value our item highly, and so we can potentially sell it for more.

2.2 Multiple-Item Model

While the single-item model is (relatively) easy to work with and analyze, it isn't always a particularly good model for what we want to do in the real world. As a motivating example, we consider auctioning off sponsored links in a Google search. If we have multiple slots in a page of results, it doesn't really make sense to auction them all off independently of one another, as an advertiser (probably) wouldn't want two slots. Thus, we wish to design a model where we can auction off k items at once, and each bidder gets at most one item.

In order to work with this model, we will again need to make some assumptions about our bidders' strategies. Importantly, we will *not* assume that the bidders value all items equally, but instead will assume the following:

Assumption 2.4. Each bidder i has some base valuation v_i, and each item j has a "quality" score q_j. If bidder i pays p for item j, their utility is $q_j v_i - p$; if they get no item, their utility is 0.

Note that this assumes that there is some objective ordering to the items — it cannot be the case that one bidder wants item 1 the best but some other bidder would prefer to have item 2.[2] In our motivating example of sponsored links, we can think of v_i as the value bidder i gets out of a click and q_j as the percentage of the time users click on link j. In this interpretation, $q_j v_i$ is bidder i's expected value if they get assigned slot j.

Our goal is now to design an "awesome" auction system for the multiple-item model. The definition of "awesome" is almost exactly the same as in Theorem 2.3; the only difference is in point (3), where we have to change the formula for social surplus. In order to ensure that the surplus corresponds to the total value the bidders get from their items, we change the constraints on the a_is to say that each q_j gets filled in for exactly one of them, while the remainder are all filled with zeros.

[2] For simplicity, we will number the items such that $q_1 \geq q_2 \geq ... \geq q_k$.

2.2.1 A General Strategy

As we discussed earlier, an auction system is effectively defined by two rules: an allocation rule that decides which items go to which bidders, and a payment rule that decides how much each bidder pays.[3] In order to design our awesome auction system, we will apply the following two step procedure:

Step 1) Assume that $b_i = v_i$ for all i. Design an efficient allocation rule that maximizes the social surplus.

Step 2) Design a payment rule that is DSIC given our allocation rule.

For our purposes, step 1 is easy enough to do: we simply assign the highest bidder to slot 1, the second highest to slot 2, and so on. This is very efficient (the slowest part is the sorting of the bids), and it is a fairly easy exercise to show that this allocation will indeed maximize the social surplus. We will spend the remainder of today focusing on step 2.

2.2.2 Myerson's Lemma

A natural question one might ask is if step 2 in our above strategy is always possible. Given any allocation rule, can we design a payment rule that will give us the DSIC property? It turns out, the answer is no. However, we are able to quantify exactly when our allocation rule admits a DSIC payment rule — and as an added bonus, we can give an explicit formula for that payment rule when it exists. To get to this, we first give the following definition.

Definition 2.5. *An allocation rule is* monotone *if, given any fixed bids from all bidders other than i, $a_i(\boldsymbol{b})$ is non-decreasing in bidder i's bid.*

Monotonicity is intuitively a reasonable property we would want from our allocation rule. After all, if a bidder could potentially get a more valuable item by bidding less, they might be incentivized to bid less than their true valuation, which is precisely what we are trying to avoid. In fact, it turns out that monotonicity is exactly the property we need from our allocation rule in order to get a DSIC payment rule, as stated in the following theorem.

Theorem 2.6 (Myerson '81)**.** *Consider an allocation rule $\boldsymbol{a}$. Then*

(a) $\boldsymbol{a}$ admits a DSIC payment rule if and only if $\boldsymbol{a}$ is monotone.

(b) If $\boldsymbol{a}$ is monotone (and assuming that $p_i(\boldsymbol{b}) = 0$ whenever $b_i = 0$), there is a unique DSIC payment rule.

(c) We can give an explicit formula for this unique payment rule.

Myerson's Lemma in fact applies to a wide variety of auction models and possible allocation functions. However, for today we will consider it in the context of our multiple-item auction model. What this will mean for us in particular is that we will only consider allocation functions which are *piecewise constant* in b_i. This means that, outside of a finite number of jumps, $a_i(\mathbf{b})$ remains constant as a function of b_i.[4]

Proof of Myerson's Lemma. Fix any bids by all bidders other than i, and let y and z be two numbers such that $0 \leq y < z$. We first consider what would happen if bidder i had true valuation z. In this case, bidder i

[3] For notational simplicity, we will often write $a_i(\mathbf{b})$ to denote the value of whatever item (if any) is allocated to bidder i given a vector of bids $\mathbf{b}$, and $p_i(\mathbf{b})$ to denote the price bidder i pays.

[4] For notational simplicity, whenever we have some fixed bids from other bidders, we will let $a_i(b_i)$ be the univariate function obtained by fixing all bids other than i in $a_i(\mathbf{b})$; we define $p_i(b_i)$ similarly for payments.

would achieve utility $z \cdot a_i(z) - p_i(z)$ by bidding z, or a utility of $z \cdot a_i(y) + p_i(y)$ by bidding y. If we want to have the DSIC property, we need to have that the bidder cannot do better by lying about their valuation, meaning we need

$$z \cdot a_i(z) - p_i(z) \geq z \cdot a_i(y) - p_i(y) \tag{2.1}$$

We can now repeat exactly the same argument in the case where y is bidder i's true valuation. This will tell us that in order to have any hope of being DSIC, we also need

$$y \cdot a_i(z) - p_i(z) \leq y \cdot a_i(y) - p_i(y) \tag{2.2}$$

We can then take these two inequalities and put all the ps on one side, giving us

$$z(a_i(z) - a_i(y)) \geq p_i(z) - p_i(y) \tag{2.3}$$

and

$$y(a_i(z) - a_i(y)) \leq p_i(z) - p_i(y) \tag{2.4}$$

Note that in particular, this must mean that

$$z(a_i(z) - a_i(y)) \geq y(a_i(z) - a_i(y)) \tag{2.5}$$

Since $z > y$, this is only possible if $a_i(z) - a_i(y) \geq 0$. Hence, we know that if we want to have any hope of making a DSIC payment rule, we need to have that $a_i(z) \geq a_i(y)$ whenever $z > y$; that is, we need our allocation rule to be monotone.

In order to complete the proof, we now just need to show that if $\mathbf{a}$ is monotone, there exists a unique DSIC payment rule (and give a formula for it). This is where we will use the fact that we are only considering piecewise constant functions. Note that by (2.3), if bids y and z get the same allocation, we must have that $p_i(z) - p_i(y) \geq 0$. Similarly, by (2.4), we must have that $p_i(z) - p_i(y) \leq 0$. Thus, if we are to have any hope of being DSIC, we must ensure that whenever $a_i(z) = a_i(y)$, $p_i(z) = p_i(y)$ as well. Since a_i is piecewise constant, we know that p_i must also be piecewise constant, with jumps only where there are jumps in a_i.

Now we consider what happens at the points where a_i jumps. We will assume that we have some point y such that the jump is just above y.[5] We fix y and consider what happens to (2.3) and (2.4) as we let z approach y from above. In this limit, the multiplicative factor of z in (2.3) will approach y, so the two equations will tell us that $p_i(z) - p_i(y)$ must be both upper and lower bounded by $y(a_i(z) - a_i(y))$, and hence must in fact equal this quantity. Putting this together with our assumption that a bid of zero results in a payment of zero, we get that the only possible DSIC payment rule for an allocation rule $\mathbf{a}$ is given by

$$p_i(b) = \sum_{j \in J_b} j \cdot \text{jump in } a_i(\cdot) \text{ at } j \tag{2.6}$$

where J_b is the set of all points no larger than b at which $a_i(\cdot)$ jumps.

Note that so far, we have only showed that no rule other than that given in (2.6) can possibly give us a DSIC auction, but in order to complete the proof, we need to show that this payment rule is actually DSIC. First, we consider what happens if bidder i bids above their valuation. For each jump point j in a_i they cross, the value of the item they get goes up by v_i times the magnitude of the jump. But at the same time, the amount they pay goes up by j times that same magnitude — so since j must be larger than v_i (as we are only considering jumps they reach moving upwards from v_i), the extra payment is larger than the extra value, so bidder i can only lose utility. Similarly, if bidder i underbids, each jump they cross below v_i decreases the amount they pay by j times the jump but also decreases the value they get by v_i times the jump, so since $j < v_i$, they lose utility. For a more visual way of understanding this, we have the following figure:

[5] The case where the jump is just below a point is almost symmetric; instead of fixing y and letting z approach y from above, we fix z and let y approach z from below.

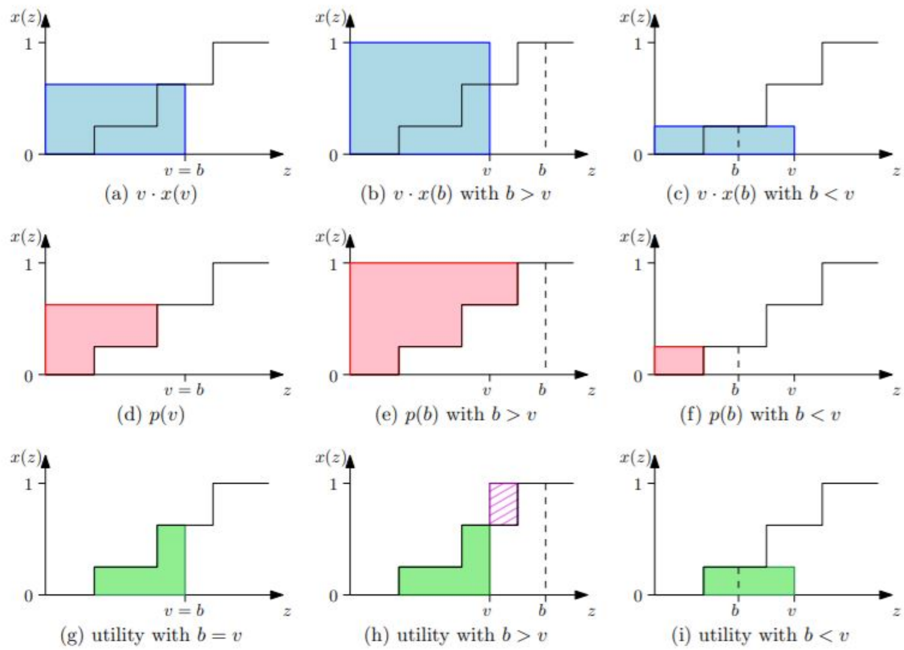

Figure 2: Proof by picture that the payment rule in (6), coupled with the given monotone and piecewise constant allocation rule, yields a DSIC mechanism. The three columns consider the cases of truthful bidding, overbidding, and underbidding, respectively. The three rows show the surplus $v \cdot x(b)$, the payment $p(b)$, and the utility $v \cdot x(b) - p(b)$, respectively. In (h), the solid region represents positive utility and the lined region represents negative utility.

Note that this figure is drawn from Tim Roughgarden's notes, and so uses slightly different notation from what we've been using here. □

2.2.3 Myerson's For Multiple-Item Auctions

Now that we have a good understanding of how to create a DSIC payment rule (and when it is possible), we will see how to apply this to our multiple-item auction model. Suppose that our k items have quality scores $q_1 \geq q_2 \geq ... \geq q_k$ and fix bids $b_1 \geq b_2 \geq ... \geq b_n$ for all the bidders.[6] All the bidders below k get an allocation of zero (the same as if they had bid nothing at all), and so must pay zero. Bidder k is above a single jump in the allocation function, from 0 to q_k. This jump occurs where they would first beat bidder $k+1$ — that is, at b_{k+1}. Hence, bidder k should pay $b_{k+1}q_k$. Bidder $k-1$ is then above this jump as well as the jump from q_k to q_{k-1}. Since this latter jump occurs at q_k, bidder $k-1$ should pay $b_{k+1}q_k + b_k(q_{k-1} - q_k)$. Continuing this pattern, we get that for $i \leq k$, bidder i will pay

$$\sum_{j=i}^{k} b_{j+1}(q_j - q_{j+1}) \tag{2.7}$$

where we define q_{k+1} to be zero; all other bidders will pay nothing, as they get allocated nothing.

[6] We may need to rename the bidders to get this order of bids.