CS 6840 Algorithmic Game Theory 11th Nov., 2024

Lecture 32: Auctions with XOS Valuations

Instructor: Eva Tardos *Scribe: Firry Yang*

1 Introduction

In this lecture, we are interested in *second-price multi-item auctions* with n bidders bidding on m items: each bidder i employs a strategy s_i which bids s_{ij} on item j, and she wins the item and pays the others' highest bid on it if she has the highest bid among all. Each bidder i has a value v_{ij} on each item j, and for the set of items that she wins, S_i, she has a valuation $v_i(S_i)$. Her resulting utility is

$$u_i(s_i, s_{-i}) = v_i(S_i) - \sum_{j \in S_i} \max_{i' \neq i} s_{i'j}. \tag{1}$$

In the previous lectures, we are mostly concerned with the simple valuation $v_i(S) = \max_{j \in S} v_{ij}$. From now on, we generalise it to a class of more complex valuations:

Definition 1 (XOS valuations)**.**

$$v_i(S) = \max_{l \in [k]} \sum_{j \in S} v_{ij}^l,$$

where the maximum is taken over the k *options*: $v^1, \ldots, v^k$.

We have seen before that any *submodular* valuation v_i is XOS by considering each of the $m!$ permutations on items, π, as an option: $v_{ij}^\pi = v_i(\{j' : \pi(j') < \pi(j)\} \cup \{j\}) - v_i(\{j' : \pi(j') < \pi(j)\})$. In fact, all XOS valuations are *subadditive*:

$$\text{submodular} \subseteq \text{XOS} \subseteq \text{subadditive}.$$

Writing $p_j := \max_i s_{ij}$ (i.e., the highest bid on item j among all bidders), we have $u_i(s_i, s_{-i}) \geq v_i(S_i) - \sum_{j \in S_i} p_j$ from (1) where S_i is the set won by i. With v_i being XOS, the following lemmas are useful for building efficient learning algorithms if the number of options k is small:

Lemma 1. *The time complexity for finding* $\arg\max_S \{v_i(S) - \sum_{j \in S} p_j\}$ *is* $O(km)$.

Lemma 2. *Given* S*, finding* $\arg\max_l \sum_{j \in S} v_{ij}^l$ *can also be done in* $O(km)$ *time.*

It gets a bit problematic if k is very large, e.g., we can have $k = m!$ options for a submodular valuation. Depending on the structure of the problem, we may still be able to complete these tasks efficiently (e.g., exploiting submodularity of the valuation).

Given that the above two tasks can be done in polynomial time, we can devise a polynomial-time *no-envy* algorithm which produces no *overbidding*. Our goal is to use this algorithm for learning so that the total *social welfare* in T iterations, $\mathsf{SW} := \sum_{\tau=1}^T \mathsf{SW}^\tau := \sum_{\tau=1}^T \sum_i v_i(S_i)$, is greater than or equal to a half of the optimal social welfare OPT minus some *regret*.

2 No-envy learning

Writing $u_i^\tau(\cdot) := u_i(\cdot, s_{-i}^\tau)$ for each bidder i, we claim that there is a polynomial-time algorithm to generate a bid s^τ which satisfies the no-envy condition:

$$\sum_{\tau=1}^{T} u_i^\tau(s^\tau) \geq \max_S \sum_{\tau=1}^{T} [v_i(S) - \sum_{j \in S} p_j^\tau] - \mathsf{Reg}. \tag{2}$$

Moreover, the algorithm forbids overbidding.

To show that this is true, we first analyse a 'cheat' algorithm where we have information about the highest bids in the following iteration, p_j^t's. At time t, we first choose

$$S^t = \arg\max_S \sum_{\tau=1}^{t} [v_i(S) - \sum_{j \in S} p_j^\tau].$$

This can be done in polynomial time because of Lemma 1.

Next, we find the option l such that $v_i(S^t) = \sum_{j \in S^t} v_{ij}^l$. This can be done in polynomial time because of Lemma 2. We output $s^t = v_i^l \equiv (v_{i1}^l, \ldots, v_{im}^l)$ as i's bidding strategy at time t. Note that there is no overbidding:

$$\forall S \ \sum_{j \in S} s_j^t = \sum_{j \in S} v_{ij}^l \leq v_i(S).$$

We claim that this 'cheat' algorithm (without overbidding) satisfies (2) with $\mathsf{Reg} = 0$.

Proof. We prove this by induction. The base case $t = 0$ is trivial.

By induction of (2), choosing $S = S^t$, we have

$$\sum_{\tau=1}^{t} u_i^\tau(s^\tau) = \sum_{\tau=1}^{t-1} u_i^\tau(s^\tau) + u_i^t(s^t) \geq \sum_{\tau=1}^{t-1} [v_i(S^t) - \sum_{j \in S^t} p_j^\tau] + u_i^t(s^t).$$

Because $p_j^t \geq s_j^t$ as p_j^t is the highest bid on j at time t,

$$u_i^t(s^t) \geq 0 \geq \sum_{j \in S^t} (s_j^t - p_j^t) = \sum_{j \in S^t} (v_{ij}^l - p_j^t) = v_i(S^t) - \sum_{j \in S^t} p_j^t,$$

which implies

$$\sum_{\tau=1}^{t} u_i^\tau(s^\tau) \geq \sum_{\tau=1}^{t} [v_i(S^t) - \sum_{j \in S^t} p_j^\tau] = \max_S \sum_{\tau=1}^{T} [v_i(S) - \sum_{j \in S} p_j^\tau]$$

by the choice of S^t.

■

For the actual algorithm, we instead choose

$$\hat{S}^t = \arg\max_S \{t v_i(S) - \sum_{\tau=1}^{t-1} \sum_{j \in S} p_j^\tau + \xi_S^t\}$$

with some non-negative random noise ξ_S^t that 'covers up' the information from p_j^t's.

We did not offer a complete proof here. To prove that this actually works, we need to define how much noise we need for the event $\{\hat{S}^t = S^t\}$ to have high probability. I.e., we need to analyse the probability that the algorithm will pick the right set even with the prices from the last time missing.

Note that if we generate i.i.d. random variables for all subsets of $[m]$, there are exponentially many of them. Instead we may generate i.i.d. random variables corresponding to the m items and use them as a 'basis' for each subset S.

3 Price of anarchy

Now we have no-envy strategies s_i^t's which also do not overbid. We are ready to show that

$$\mathsf{SW} \geq \frac{1}{2}\mathsf{OPT} - O(n)\mathsf{Reg}.$$

Proof. Using S_i^* to denote the set of items won by i at the social optimum, by the no-envy condition (2) we have:

$$\mathsf{SW} \geq \sum_{i,\tau} u_i(s_i^\tau, s_{-i}^\tau) \geq \sum_{i,\tau} v_i(S_i^*) - \sum_{i,\tau}\sum_{j\in S_i^*} p_j^\tau - n\mathsf{Reg} = \mathsf{OPT} - \sum_\tau \sum_j p_j^\tau - n\mathsf{Reg}.$$

From no overbidding, we have for each τ:

$$\sum_j p_j^\tau = \sum_i \sum_{j\in S_i^\tau} p_j^\tau \leq \sum_i v_i(S_i^\tau) = \mathsf{SW}^\tau$$

by decomposing the m items into the set won by each bidder i at time τ, S_i^τ.

Therefore,

$$\mathsf{SW} \geq \mathsf{OPT} - \sum_\tau \mathsf{SW}^\tau - n\mathsf{Reg} = \mathsf{OPT} - \mathsf{SW} - n\mathsf{Reg},$$

which rearranges to

$$\mathsf{SW} \geq \frac{1}{2}\mathsf{OPT} - \frac{n}{2}\mathsf{Reg}.$$

■

This is the same bound as we apply no-envy learning in auctions with the simple valuation $v_i(S) = \max_{j\in S} v_{ij}$. Similarly to what we have seen in first-price auctions, this is another generalisation of the (λ, μ)-*smooth* results with single-item auctions to multi-item auctions with XOS valuations.