princeton univ. F'17 cos 521: Advanced Algorithm Design

Lecture 19: Combinatorial Auctions II

Lecturer: *Matt Weinberg* Scribe:*Matt Weinberg*

Lectures Notes sourced from the Chapter 12 of the AGT Book (Nisan, Roughgarden, Tardos, Vazirani), cited at end. I strongly recommend visiting the source, but these lecture notes are available just to recall what was covered in lecture.

Like last class, we'll continue taking a game-theoretic look at algorithmic problems, and focus on combinatorial auctions. There are n bidders and m items. Each bidder has a *valuation function* for the m items $v_i(\cdot)$. That is, player i receives value $v_i(S)$ for receiving the set S of items. Your optimization problem is to find an allocation, that is a partition of $[m]$ into $S_1, \ldots, S_n$ so as to maximize the *social welfare*, $\sum_i v_i(S_i)$.

Last class, we saw a generic reduction that took as input an algorithm that maximized welfare without incentives, and turned it into a dominant strategy truthful mechanism called Vickrey-Clarke-Groves. We also saw a DST approximation algorithm when each $v_i(\cdot)$ was single-minded. This class, we'll see a DST approximation algorithm for arbitrary monotone valuations, but the mechanism will be randomized.

1 Lavi-Swamy Reduction

The reduction is based on tools we've seen throughout class, LP relaxations and rounding. The high-level approach is the following. First, we'll define a class of mechanisms that are "VCG-like," where similar reasoning to last class lets us turn algorithms of a certain form into DST mechanisms.

1.1 Maximal-In-Range and Maximal-In-Distributional-Range

Let's first recall what the VCG reduction did. We took as input any algorithm $\mathcal{A}$ that on input v selected the welfare-maximizing partition $\mathcal{A}(v)$. That is, $\mathcal{A}(v)$ maximizes $\sum_i v_i(\mathcal{A}_i(v))$ over all partitions. Then, we charged each bidder their "externality," how much the other bidders' welfare went down due to their presence: $\sum_{j \neq i} v_j(\mathcal{A}_j(v_1, \ldots, v_{i-1}, 0, v_{i+1}, \ldots, v_n)) - \sum_{j \neq i} v_j(\mathcal{A}_j(v))$. We then observed that bidder i's utility (value minus price) turned out to be exactly the total welfare minus a term completely out of their control, so bidder i always wants to maximize the welfare (which can be achieved by reporting their true value). Finally, we concluded that because $\mathcal{A}$ maximized welfare on all inputs, that VCG also maximizes welfare on all inputs.

Now we make a key observation: with the VCG payment rule, for any algorithm $\mathcal{A}$, the resulting mechanism is DST if and only if for all v, v' where $v_j = v'_j$ for all $j \neq i$, $\sum_j v_j(\mathcal{A}_j(v)) \geq \sum_j v_j(\mathcal{A}_j(v'))$. That is, given that we're going to run algorithm $\mathcal{A}$, and the other bidders have already reported v_{-i}, welfare is maximized when bidder i reports value v_i. This might seem like an extremely natural property that all approximation algorithms should have, but surprisingly it's not (see homework). But if we use an algorithm $\mathcal{A}$ with this property, then the VCG mechanism using $\mathcal{A}$ is still DST. Only the last sentence in the paragraph above isn't necessarily true: $\mathcal{A}$ may no longer maximize welfare on all inputs.

As an example, consider $\mathcal{A}$ that always gives all items to the same bidder. Then the VCG mechanism essentially becomes a second-price auction for the grand bundle $[m]$ of all items, and is DST. But it does a horrible job of maximizing welfare because it never considers splitting the items.

Such mechanisms are called *maximal-in-range*: there exists some set S of possible allocations, and $\mathcal{A}_S(v)$ outputs the best allocation in S (the one that maximizes welfare for v). An extension of these ideas is called *maximal-in-distributional-range* and instead lets S be a set of *distributions over allocations* (or *randomized allocations*). Again, $\mathcal{A}_S(v)$ outputs the welfare-maximizing distribution in S.

For example, maybe S contains all distributions that pick two different bidders for each item, then allocates each item independently to one of the two bidders uniformly at random. Then $\mathcal{A}_S(v)$ would find, over all such $m^{\binom{n}{2}}$ (randomized) allocations the one that maximizes expected welfare (formally: maximizes $\mathbb{E}[\sum_i v_i((\mathcal{A}_S)_i(v))]$). I'm not claiming that this is an interesting class of distributions to optimize over, but it fits the definition.

The remaining goal of this lecture will be to design a maximal-in-distributional-range mechanism that guarantees a good approximation.

1.2 The Configuration LP

Our maximal-in-distributional-range algorithm will use a specific LP relaxation (also used in many other problems) called the configuration LP. It is as follows. Below, think of $x_{i,S}$ as the fraction of set S awarded to bidder i.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i,S} x_{i,S} v_i(S) \\
\text{subject to} \quad & \sum_i \sum_{S \ni j} x_{i,S} \le 1 \quad \forall j \in [m] \\
& \sum_S x_{i,S} \le 1 \quad \forall i \in [n] \\
& x_{i,S} \ge 0 \quad \forall i \in [n], S \subseteq [m]
\end{aligned}$$

First observe that this LP is indeed a relaxation: for any allocation $S_1, \ldots, S_n$, we can set $x_{i,S_i} = 1$, and all other variables equal to zero. Observe also that, unfortunately, it has exponentially many variables. That's a shame, so we'll have to be clever in order to solve the LP in poly-time. Fortunately, there aren't too many constraints, so the dual has few variables and exponentially many constraints - this is starting to sound closer to something we can solve. The dual is as follows:

$$\begin{aligned}
\text{minimize } & \sum_i u_i + \sum_j p_j \\
\text{subject to } & u_i \geq v_i(S) - \sum_{j \in S} p_j \quad \forall i, S \\
& u_i, p_j \geq 0
\end{aligned}$$

We can solve the dual as long as we can get a poly-time separation oracle. Whether or not this is possible depends exactly on how each $v_i(\cdot)$ is represented. We'll assume its represented in a way so that we can evaluate a *demand query*. That is, we can take as input a vector of price $p_1, \ldots, p_m$ and output $\arg\max_S\{v_i(S) - \sum_{j \in S} p_j\}$. Notice that this is bidder i's favorite set if the items are priced at $p_1, \ldots, p_m$. It's normally considered a reasonable kind of query, since bidder i themselves is presumably capable of picking their favorite set at some prices in poly-time (philosophical aside: if not, then maybe you should redefine v_i to match whatever bidder i would select?).

Observation 1
With demand query access to each $v_i(\cdot)$, we can implement a poly-time separation oracle.

Proof: Simply execute a demand query for each v_i at prices $\vec{p}$. If u_i exceeds the resulting $v_i(S) - \sum_{j \in S} p_j$, then all constraints for bidder i are satisfied. If not, we've explicitly found a violated constraint. $\Box$

For the rest of this class, we'll assume demand query access to the valuations, and therefore we can get a separation oracle for the dual and solve the configuration LP in poly-time. Note that it's not trivial to take a solution to the dual and transform it into a solution for the primal, but it's not too hard (it uses complementary slackness). Notice though that because the primal only has $n + m$ constraints, there always exists an optimal solution where only $n + m$ coordinates are non-zero (this is because there always exists an optimal solution that is a corner, and all but $n + m$ tight constraints at the corner are setting some variable to zero). So now that we can solve the LP relaxation, we have to figure out what to do with it.

1.3 Rounding the Configuration LP

Definition 1 *We say that a rounding algorithm $\mathcal{A}$ verifies an integrality gap of $c \leq 1$ for the configuration LP if it takes as input v and outputs a (deterministic) allocation such that $\sum_i v_i(\mathcal{A}_i(v)) \geq c \cdot \text{ConfigOPT}(v)$, where $\text{ConfigOPT}(v)$ is the fractional optimum of the configuration LP.*

Proposition 1
[Lavi/Swamy 2005] Let $\mathcal{A}$ verify an integrality gap of c for the configuration LP. Then for any fractional solution x to the configuration LP with k non-zero coordinates, one can decompose $c \cdot x$ into a distribution over integral allocations in poly(n, m, k) black-box calls to $\mathcal{A}$ (and poly(n, m, k) overhead).

Proof: (**Note: this was omitted in lecture)** Strongly recommend visiting Chapter 12 of the cited textbook for this since there are some subtleties, but the main ideas are below). Consider the following LP, which tries to write $c \cdot x$ as a convex combination of (exponentially many) integral allocations (let $\mathcal{I}$ denote the set of integral allocations, which is finite):

$$\begin{aligned} \text{minimize } & \sum_{y \in \mathcal{I}} \lambda_y \\ \text{subject to } & \sum_{y \in \mathcal{I}} \lambda_y x^y_{i,S} = c \cdot x_{i,S} \quad \forall (i,S) \text{ such that } x_{i,S} > 0 \\ & \sum_{y \in \mathcal{I}} \lambda_y \geq 1 \qquad \text{(this constraint might seem silly, but it's helpful to keep the dual clean)} \\ & \lambda_y \geq 0 \end{aligned}$$

Quickly observe that we don't need to enforce the constraint $\sum_{y \in \mathcal{I}} \lambda_y x^y_{i,S} = 0$ when $x_{i,S} = 0$, we can simply ignore any integral allocations that award set S to bidder i when $x_{i,S} = 0$ (e.g. hard-code such $\lambda_y = 0$, or just remove these variables entirely). Finally, also observe that if we find a solution to this LP where $\sum_{y \in \mathcal{I}} \lambda_y = 1$, this is exactly a convex combination over integral allocations. So our goal is to find such a solution, via the dual:

$$\begin{aligned} \text{maximize } z + & \sum_{(i,S), x_{i,S} > 0} c x_{i,S} \cdot w_{i,S} \\ \text{subject to } z + & \sum_{(i,S), x_{i,S} > 0} x^y_{i,S} w_{i,S} \leq 1 \quad \forall y \in \mathcal{I} \\ z & \geq 0 \end{aligned}$$

We now want to claim that we can solve the dual in the desired runtime, given black-box access to $\mathcal{A}$, and that the value of the dual must be 1. First, observe that $z = 1, w_{i,S} = 0$ for all (i,S) is a valid dual solution and has value 1. So the dual has value at least one (this is why it was helpful to write the superfluous constraint in the primal, although we could have drawn the same conclusions without it and a little extra reasoning. Also observe that the dual has only $k+1$ variables, so if we can get a separation oracle, we can solve it in time poly(k).

Now we want to show that the dual has no feasible solutions with value > 1. Assume for contradiction that (w, z) was such a solution. Then consider running algorithm $\mathcal{A}$ on input valuations with $v_i(S) = w_{i,S}$. Then this produces some integral allocation y with $\sum_{i,S} x^y_{i,S} w_{i,S} \geq c \sum_{i,S} x_{i,S} w_{i,S} > 1 - z$. The last inequality follows from hypothesis that we started with a feasible solution of value > 1. But now we can rearrange this into a violated constraint in the dual, and therefore the solution was in fact not feasible.

There are some subtleties this time for recovering the optimal primal from the optimal dual, but we'll again omit this. So we can again recover a solution to the primal with only k integral allocations, and this is the convex combination we desire.

□

1.4 Verifying the Integrality Gap

Finally, we'll observe (without proof, it's similar to last class, check the cited book chapter for a proof) that the same greedy algorithm from last class verifies an integrality gap of $1/\sqrt{2m}$. That is, on input v, sort all (i, S) in decreasing order of $v_i(S)/\sqrt{|S|}$, then greedily assign sets that don't conflict (but now two sets conflict if they're of the same bidder as well).

1.5 Putting everything together

So the complete picture looks like this:

1. Let $\mathcal{A}$ denote the algorithm that decomposes a point $\frac{1}{\sqrt{2m}} \cdot x$, where x is feasible for the configuration LP into a distribution over integral allocations. This is based on the greedy algorithm, plugged through Proposition 1.

2. Let S denote the set of all distributions that $\mathcal{A}$ might ever output on input x, where x is feasible for the configuration LP. Let $\mathcal{B}$ be the algorithm that on input v, solves the configuration LP, then runs $\mathcal{A}$ to get a distribution over integral allocations. Note that $\mathcal{B}$ finds the welfare maximizing distribution in S.

3. Use the VCG payments with maximal-in-distributional-range algorithm $\mathcal{B}$. This is a DST mechanism. Observe that it guarantees a $\frac{1}{\sqrt{2m}}$-approximation.

Bibliography

1. Algorithmic Game Theory. Nisan, Roughgarden, Tardos, Vazirani (eds.), Cambridge University Press 2007.