Algorithmic Game Theory

LECTURE 4

Main Topics Covered:

- Knapsack Auctions
- Algorithmic Mechanism Design
- Algorithmic Mechanism Design for Knapsack Auctions
- Revelation Principle

Knapsack Auctions (1/5)

- An example of single-parameter environments

SETUP:

- Each bidder i has
  - A publicly known size w_i
  - A private valuation v_i
- Seller has capacity = W
- Feasible set X is the set of 0-1 vectors $(x_1, x_2, \ldots, x_n)$ such that $\sum_{i=1}^{n} w_i x_i \leq W$

<u>EXAMPLE :- DISPLAYING A TELEVISION AD</u>

- Bidder size = Duration of company's Ad
- Bidder valuation = Price he is willing to pay for displaying the Ad
- Seller capacity = Duration of a commercial break

Knapsack Auctions (2/5)

- Some more examples:
  - A k-unit auction with $w_i = 1$ and $W = k$
  - Bidders wanting to store files on a shared server
  - Data streams sent through a shared communication channel
  - Processes to be executed on a shared supercomputer

Now,
Let's design an ideal auction using the 2-step design paradigm [Lecture 2]

1. Assume without justification that bids are truthful and design allocation rule
2. Design a payment rule s.t. our allocation rule extends to a DSIC mechanism

Knapsack Auctions (3/5)

STEP 1:

- Assume → Bids are truthful i.e. $b_i = v_i$!
- For Surplus Maximization, we choose a bidder out of X who can maximise the surplus

  i.e. allocation rule ➔ $x(b) = argmax_X \sum_{i=1}^{n} b_i x_i$
- In other words, we need to solve a 'knapsack problem' where
  - Item values = Reported bids $(b_1,b_2,\ldots,b_n)$
  - Item sizes = Bidder sizes $(w_1,w_2,\ldots,w_n)$

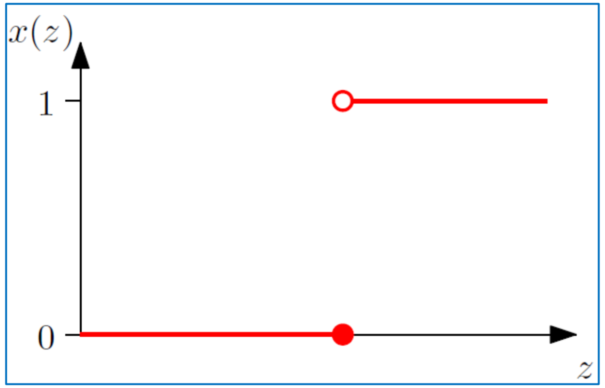
In general, all single-parameter environments surplus maximization leads to a "monotone" allocation rule !

Therefore,
We can now use Myerson's Lemma to get our payment rule in STEP 2

Knapsack Auctions (4/5)

STEP 2:

- Fix a bidder i and bids b_{-i} by other bidders
- The 0-1 monotone allocation rule gives an allocation curve $x_i(\cdot, b_{-i})$ that is 0 until some breakpoint z at which it jumps to 1



Using Myerson's payment formula [Lecture 3] :

$$p_i(b_i, \boldsymbol{b}_{-i}) = \sum_{j=1}^{l} z_j.\,[jump\ in\ x_i(\cdot, b_{-i})\ at\ z_j]$$

Where, $(z_1, z_2, \ldots, z_l)$ are the breakpoints, we conclude that

1. If i bids < z → i loses and pays 0
2. If i bids > z → i wins and pays z.(1-0)=z

We call 'z' the "Critical Bid" – the infimum bid that bidder can make and still win

Thus, the winner has to pay his/her critical bid !

Knapsack Auctions (5/5)

RECALL : For an auction to be ideal it has to be 1) DSIC, 2)surplus maximising and 3)computationally efficient [Lecture 2]

Q: So is our mechanism ideal ?

NO !

A: Because knapsack problem is NP-Hard ! (i.e. there is no poly-time implementation of our allocation rule unless P=NP)

Possible Solution:

We can relax any one of the 3 constraints for being an ideal auction

- Relaxing 1 is of no help as 2 & 3 are still not compatible
- On relaxing 3, we can now implement the allocation rule in a pseudo-polynomial time using dynamic programming
- In this lecture, however, we choose to relax 2 and keep 1 & 3 !

  [This is where Algorithmic Mechanism Design comes in]

Algorithmic Mechanism Design

- Deals with the issue of relaxing the 2nd requirement (surplus maximization) of Ideal auctions as little as possible subject to the 1st (DSIC) and 3rd (poly-time) requirements
- ***Equivalently:***

  Myerson's lemma reduces this task further to the design of a poly-time and 'monotone' allocation rule that comes as close as possible to surplus maximization
- NOTE:

  Algorithmic mechanism design is very similar to *approximation algorithms* (whose goal is to design poly-time algorithm for NP-hard problems that are as close as possible to optimal)
- Best-case Scenario:

  Where we design a poly-time allocation rule just like approximation algorithms and it turns out to be 'monotone' as well !

With this knowledge of algorithmic mechanism design in mind, let's come back to our knapsack auction problem

Algorithmic Mechanism Design for Knapsack Auctions (1/2)

- We can use a Greedy approach but what can be the basis ?
- Our requirements:
  - A bigger bid b_i looks good [since bigger bid means bigger surplus $(\sum_{i=1}^{n} b_i x_i)$ and hence better maximization]
  - A smaller size w_i looks good [since the lesser the size the better, given we have a seller capacity W]
- The Greedy Knapsack Heuristic is as follows:

1. Sort and re-index the bidders so that $\frac{b_1}{w_1} \geq \frac{b_2}{w_2} \geq \cdots \geq \frac{b_n}{w_n}$
2. Pick winners in this order until one doesn't fit and then halt

   [e.g., when the capacity left is 3 and we have our next bidder with size 4, we stop]
3. Return the solution from the previous step or the highest bidder, whichever has larger social welfare

Algorithmic Mechanism Design for Knapsack Auctions (2/2)

- NOTE: In step 2, we have an issue !
  - What if the capacity left is 3, we encounter a bidder with size 4 and halt but later there is a bidder with size 2 !
  - So, to resolve this, alternatively, instead of halting we can continue following the sorted order until some bidder fits
- ***Quick Fact 1:***
  - Assuming truthful bids, the social welfare achieved by this greedy allocation rule is at least 50% of the maximum social welfare

> ❑ Note →
> The percentage can be improved even further if we assume that the bidder sizes are not too big or more precisely a sufficient enough fraction of the Knapsack having size W

- ***Quick Fact 2:***
  - This greedy allocation rule is also monotone and by Myerson's lemma it is also DSIC !

Revelation Principle (1/3)

Introduction →

- Till now we have studied only DSIC mechanisms
- Recall:

  The 2 conditions for DSIC were:

  - For every valuation profile, the mechanism has a 'dominant strategy equilibrium' – an outcome that results from every participant playing a dominant strategy
  - In this dominant strategy equilibrium, every participant truthfully reports her private valuation to the mechanism
- The Revelation Principle states that given requirement (1), requirement (2) comes for free !
- Formally:

Revelation Principle for DSIC mechanism →
For every mechanism M in which every participant always has a dominant strategy, there is an equivalent direct-revelation DSIC mechanism M´

Revelation Principle (2/3)

Proof of Revelation Principle →

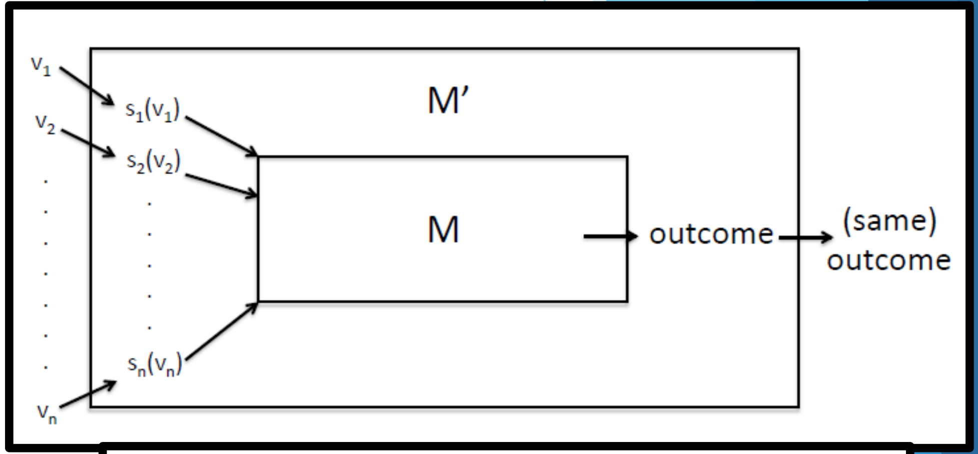
- Assume:
  For every participant i with private information v_i, i has a dominant strategy $s_i(v_i)$ in given mechanism M

- Construct a new mechanism M´ to which participants give the responsibility of playing the appropriate dominant strategy

- M´ accepts sealed bids $b_1,\ldots,b_n$!

- M´ submits bids $s_1(b_1),\ldots,s_n(b_n)$ to M and chooses the same outcome as M

- If a participant i submits a bid other than v_i then M´ ends up submitting something other than $s_i(v_i)$ to M, thereby reducing i's utility



Construction of direct revelation mechanism M´ given a dominant strategy mechanism M

Revelation Principle (3/3)

- Question →

  Can we obtain better mechanisms that do not have a dominant strategy ?
- Issues with such a mechanism:
  - Difficult to predict what bidders will do
  - Difficult to predict the outcome of the mechanism
- Answer →

  Sometimes YES! , but not always
- DSIC mechanisms provide better incentive guarantees
- Non-DSIC mechanisms provide better performance guarantees and prove to be useful in complex problems

Conclusion → Use of DSIC or non-DSIC mechanisms depends on the problem !

References

- Twenty Lectures on Algorithmic Game Theory by Tim Roughgarden, 2016, Cambridge University Press

- https://www.youtube.com/playlist?list=PLEGCF-WLh2RJBqmxvZ0_ie-mleCFhi2N4

THANK YOU