CS364A: Algorithmic Game Theory
Lecture #4: Algorithmic Mechanism Design*

Tim Roughgarden†

October 2, 2013

1 Knapsack Auctions

Next we design DSIC mechanisms for *knapsack auctions*. These will be single-parameter environments, so Myerson's Lemma will apply.

1.1 Problem Definition

In a knapsack auction, each bidder i has a publicly known *size* w_i (e.g., the duration of a TV ad) and a private valuation (e.g., a company's willingness-to-pay for its ad being shown during the Super Bowl). The seller has a capacity W (e.g., the length of a commercial break). The feasible set X is defined as the 0-1 n-vectors $(x_1, \ldots, x_n)$ such that $\sum_{i=1}^{n} w_i x_i \leq W$. (As usual, $x_i = 1$ indicates that i is a winning bidder.) Other situations such knapsack auctions model include bidders who want files stored on a shared server, data streams sent through a shared communication channel, or processes to be executed on a shared supercomputer. (When there is a shared resource with limited capacity, you have a Knapsack problem.) Note that k-item auctions (k identical copies of a good, one per customer) is the special case where $w_i = 1$ for all i and $W = k$. Here, different bidders can have different sizes.

Let's try to design an awesome auction using our two-step design paradigm. Recall that we first assume without justification that bids equal values and then decide on our allocation rule. Then we pay the piper and devise a payment rule that extends the allocation rule to a DSIC mechanism.

†Department of Computer Science, Stanford University, 462 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

1.2 A Surplus-Maximizing DSIC Mechanism

Since awesome auctions are supposed to maximize surplus, the answer to the first step is clear: define the allocation rule by

$$\mathbf{x}(\mathbf{b}) = \arg\max_{X} \sum_{i=1}^{n} b_i x_i. \tag{1}$$

That is, the allocation rule solves an instance of the Knapsack problem[1] in which the item (i.e., bidder) values are the given bids $b_1, \ldots, b_n$, and the item sizes are the a priori known sizes $w_1, \ldots, w_n$. By definition, when bidders bid truthfully, this allocation rule maximizes the social surplus.

1.3 Critical Bids

Myerson's Lemma (parts (a) and (b)) guarantees the existence of a payment rule $\mathbf{p}$ such that the mechanism $(\mathbf{x}, \mathbf{p})$ is DSIC. This payment rule is easy to understand. Fix a bidder i and bids $\mathbf{b}_{-i}$ by the other bidders. Since the allocation rule is monotone and 0-1, the allocation curve $x_i(\cdot, \mathbf{b}_{-i})$ is extremely simple: it is 0 until some breakpoint z, at which point it jumps to 1 (Figure 1). Recall the payment formula

$$p_i(b_i, \mathbf{b}_{-i}) = \sum_{j=1}^{\ell} z_j \cdot \text{jump in } x_i(\cdot, \mathbf{b}_{-i}) \text{ at } z_j, \tag{2}$$

where $z_1, \ldots, z_\ell$ are the breakpoints of the allocation function $x_i(\cdot, \mathbf{b}_{-i})$ in the range $[0, b_i]$ Thus, if i bids less than z, it loses and pays 0. If i bids more than z, it pays $z \cdot (1 - 0) = z$. That is, i pays its *critical bid* — the lowest bid it could make and continue to win (holding the other bids $\mathbf{b}_{-i}$ fixed). Note this is exactly what's going on in the Vickrey auction.

1.4 Intractability of Surplus Maximization

The mechanism proposed in Section 1.2 maximizes social surplus, assuming truthful bids — an assumption justified by its DSIC property. The mechanism thus solves the surplus-maximization problem with unknown data (the v_i's) as well as if this data was known a priori. But is the mechanism awesome in the sense of the Vickrey auction (Lecture 2)? Recall this means:

(1) DSIC.

(2) Surplus-maximizing, assuming truthful bids.

[1] An instance of the Knapsack problem is defined by $2n + 1$ numbers: item values $v_1, \ldots, v_n$, item sizes $w_1, \ldots, w_n$, and a knapsack capacity W. The goal is to compute the subset of items of maximum total value that has total size at most W. See any undergraduate textbook for more details.

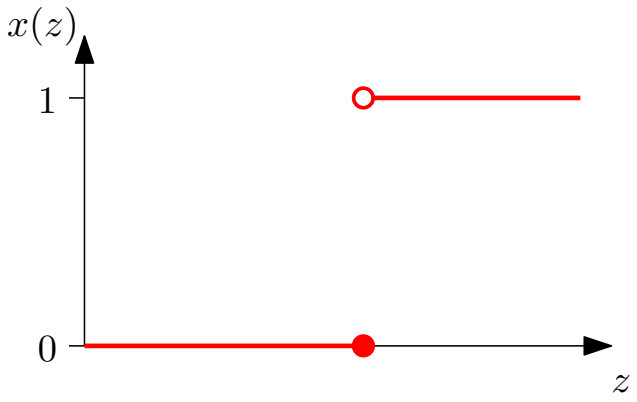

Figure 1: A monotone 0-1 allocation rule.

(3) Runs in polynomial time.

The answer is *no*. The reason is that the Knapsack problem is NP-hard. Thus, there is no polynomial-time implementation of the allocation rule in (1), unless $P = NP$. Thus, properties (2) and (3) are incompatible.

The fact that there is no awesome knapsack auction (assuming $P \neq NP$) motivates relaxing at least one of the above three goals. But which one? First, note that relaxing the DSIC condition will not help at all, since it is the second two properties that conflict.

A perfectly valid approach, which won't get much airtime in this course, is to relax the third constraint. This is particularly attractive for knapsack auctions, since the allocation rule (1) can be implemented in pseudopolynomial time using dynamic programming (again, see any undergraduate algorithms textbook for details). More generally, if your instances are small or structured enough and you have enough time and computing power to implement optimal surplus-maximization, by all means do so — the resulting allocation rule is monotone and can be extended to a DSIC mechanism.[2]

2 Algorithmic Mechanism Design

2.1 The Holy Grail: DSIC For Free

Algorithmic mechanism design is one of the initial and most well-studied branches of algorithmic game theory, and we won't have time to do it justice. The dominant paradigm in algorithmic mechanism design is to relax the second constraint (optimal surplus) as little as possible, subject to the first (DSIC) and third (polynomial-time) constraints. For single-parameter environments, Myerson's Lemma implies that the following goal is equivalent: design a polynomial-time and monotone allocation rule that comes as close as possible to maximizing the social surplus.

[2]Be warned, though, that the payments also need to be computed, which generally requires solving n more surplus-maximization problems (one per player). See also Exercise 16.

One of the reasons there has been so much progress in algorithmic mechanism design over the past 15 years is that, on a technical level, it bears a strong resemblance to the relatively mature field of *approximation algorithms*. The primary goal in approximation algorithms is to design algorithms for NP-hard problems that are as close to optimal as possible, subject to a polynomial-time constraint. Algorithmic mechanism design (for single-parameter problems) has *exactly* the same goal, except the algorithms must additionally obey a monotonicity constraint. Myerson's Lemma implies that algorithmic mechanism design boils down to algorithm design in an oddly restricted (via monotonicity) "computational model" — the entire game-theoretic aspect of the design goal is neatly compiled into a relatively intuitive extra constraint on the allocation rule.

It should be clear what the "holy grail" in algorithmic mechanism design is: for as many NP-hard problems of interest as possible, to match the best-known approximation guarantee for (not necessarily monotone) approximate surplus maximization algorithms — or even the best-possible approximation guarantee, subject to $P \neq NP$. That is, we would like the DSIC/monotone constraint to cause no additional surplus loss, beyond the loss we already have to suffer due to the polynomial-time constraint. Recall that we've been spoiled so far: with *exact* surplus-maximization, the DSIC/monotone constraint is satisfied "for free", and exact surplus-maximization with unknown data reduces to exact surplus-maximization with known data. Does an analogous reduction hold for *approximate* surplus maximization?

2.2 Knapsack Auctions (Reprise)

To explore the question above in a concrete setting, let's return to knapsack auctions. There are a number of heuristics for the knapsack problem that have good worst-case performance guarantees. For example, consider the following allocation rule $\mathbf{x}^G$, which given bids $\mathbf{b}$ chooses a feasible set — a set S of winners with total size $\sum_{i \in S} w_i$ at most the capacity W — via a simple greedy algorithm. We assume, without loss of generality, that $w_i \leq W$ for every i (it is harmless to delete bidders i with $w_i > W$).

(1) Sort and re-index the bidders so that[3]

$$\frac{b_1}{w_1} \geq \frac{b_2}{w_2} \geq \cdots \geq \frac{b_n}{w_n}.$$

(2) Pick winners in this order until one doesn't fit, and then halt.[4]

(3) Return either the step-2 solution, or the highest bidder, whichever creates more surplus.[5]

[3] What makes a bidder attractive is a high bid and a small size. We trade these off by ordering bidders by "bang-per-buck" — the value contributed per unit of capacity used.

[4] One can also continue to follow the sorted order, packing any further bidders that happen to fit — this will only do better.

[5] The reason for this step is that the solution in step 2 might be highly suboptimal if there a very valuable and very large bidder. One can also sort the bidders in nondecreasing bid order and pack them greedily — this will only do better.

The above greedy algorithm is a $\frac{1}{2}$-approximation algorithm for the Knapsack problem, which gives the following guarantee.

Theorem 2.1 *Assuming truthful bids, the surplus of the greedy allocation rule is at least 50% of the maximum-posible surplus.*

Proof: (Sketch.) Consider truthful bids $v_1, \ldots, v_n$, known sizes $w_1, \ldots, w_n$, and a capacity W. Suppose, as a thought experiment, we relax the problem so that a bidder can be chosen *fractionally*, with its value pro-rated accordingly. For example, if 70% of a bidder with value 10 is chosen, then it contributes 7 to the surplus. Here is a greedy algorithm for this "fractional knapsack problem": sort the bidders as in step (1) above, and pick winners in this order until the entire capacity is fully used (picking the final winner fractionally, as needed). A straightforward exchange argument proves that this algorithm maximizes the surplus over all feasible solutions to the fractional knapsack problem.

Suppose in the optimal fractional solution, the first k bidders in the sorted order win and the $(k+1)$th bidder fractionally wins. The surplus achieved by steps (1) and (2) in the greedy allocation rule is exactly the total value of the first k bidders. The surplus achieved in step (3) in the greedy allocation rule is at least the total value of the $(k+1)$th bidder. The better of these two solutions is at least half of the surplus of the optimal fractional solution, which is at least the surplus of an optimal (non-fractional) solution to the original problem. ■

The greedy allocation rule is even better under additional assumptions. For example, if $w_i \leq \alpha W$ for every bidder i, with $\alpha \in (0, \frac{1}{2}]$, then the approximation guarantee improves to $1 - \alpha$, even if the third step of the algorithm is omitted.

We know that surplus-maximization yields a monotone allocation rule; what about *approximate* surplus-maximization? At least for the greedy allocation rule above, we still have monotonicity (Exercise 18).

You may have been lulled into complacency, thinking that every reasonable allocation rule is monotone. The only non-monotone rule we've seen in the "second-highest bidder wins" rule for single-item auctions, which we don't care about anyways. *Warning:* natural allocation rules are not always monotone. For example, for every $\epsilon > 0$, there is a $(1-\epsilon)$-approximation algorithm for the Knapsack problem that runs in time polynomial in the input and $\frac{1}{\epsilon}$ — a "fully polynomial-time approximation scheme (FPTAS)". The rule induced by the standard implementation of this algorithm is *not* monotone, although it can be tweaked to restore monotonicity without degrading the approximation guarantee (see the Problems for details). This is characteristic of work in algorithmic mechanism design: consider an NP-hard optimization problem, check if the state-of-the-art approximation algorithm directly leads to a DSIC mechanism and, if not, tweak it or design a new approximation algorithm that does, hopefully without degrading the approximation guarantee.

2.3 Black-Box Reductions

Algorithmic mechanism design has been extremely successful for the single-parameter problems we've been discussing so far. The state-of-the-art approximation algorithms for such problems are generally either monotone or can redesigned to be monotone, like in the case of knapsack auctions mentioned above and in the problems. This success has been so widespread as to suggest the question:

> Is there a natural single-parameter problem for which the best approximation guarantee achievable by a polynomial-time algorithm is strictly better than the best approximation guarantee achievable by a polynomial-time *and monotone* algorithm?

Of course, a negative answer would be especially exciting — it would imply that, as with exact surplus-maximization, the monotonicity/DSIC constraint can always be added "for free". One way of proving such a sweeping positive result would be via a "black-box reduction": a generic way of taking a possibly non-monotone polynomial-time algorithm and transmuting it into a monotone polynomial-time algorithm without degrading the approximation guarantee. Such a reduction would be very interesting even if the approximation factor suffered by a constant factor.

Recent work by Chawla et al. [1] shows that there is no fully general black-box reduction of the above type for single-parameter environments. There might well be large and important subclasses of such environments, though, for which a black-box reduction exists. For example, does such a reduction apply to all *downward-closed* environments where, like in all of our examples so far, giving a bidder less stuff cannot render an outcome infeasible?[6]

3 The Revelation Principle

3.1 The DSIC Condition, Revisited

To this point, our mechanism design theory has studied only DSIC mechanisms. We reiterate that there are good reasons to strive for a DSIC guarantee. First, it is easy for a participant to figure out what to do in a DSIC mechanism: just play the obvious dominant strategy. Second, the designer can predict the mechanism's outcome assuming only that participants play their dominant strategies, a relatively weak behavioral assumption. Nevertheless, non-DSIC mechanisms like first-price auctions can also be useful in practice.

Can non-DSIC mechanisms accomplish things that DSIC mechanisms cannot? To answer this question, let's tease apart two separate assumptions that are conflated in our DSIC definition:

(1) Every participant in the mechanism has a dominant strategy, no matter what its private valuation is.

[6] If the DSIC constraint is weakened to an implementation in Bayes-Nash equilibrium, then there are quite general black-box reductions. We'll discuss these in more advanced lectures.

(2) This dominant strategy is *direct revelation*, where the participant truthfully reports all of its private information to the mechanism.

There are mechanisms that satisfy (1) but not (2). To give a silly example, imagine a single-item auction in which the seller, given bids $\mathbf{b}$, runs a Vickrey auction on the bids $2\mathbf{b}$. Every bidder's dominant strategy is then to bid half its value.

3.2 Beyond Dominant-Strategy Equilibria

Suppose we relax condition (1). The drawback is that we then need stronger assumptions to predict the behavior of participants and the mechanism's outcome; for example, we might consider a Bayes-Nash equilibrium with respect to a common prior (see Problem 6 on first-price auctions) or a Nash equilibrium in a full-information model (see Problem 3 on the GSP sponsored search auction). But if we're willing to make such assumptions, can we do better than with DSIC mechanisms?

The answer is "sometimes, yes." For this reason, and because non-DSIC mechanisms are common in practice, it is important to develop mechanism design theory beyond DSIC mechanisms. We'll do this in more advanced lectures. A very rough rule of thumb is that, for sufficiently simple problems like those in our introductory lectures, DSIC mechanisms can do anything non-DSIC mechanisms can. In more complex problems, like some discussed in the advanced lectures, weakening the DSIC constraint (e.g., to implementation in Bayes-Nash equilibrium) often allows you accomplish things that are provably impossible for DSIC mechanisms (assuming participants figure out and coordinate on the desired equilibrium). DSIC and non-DSIC mechanisms are incomparable in such settings — the former enjoy stronger incentive guarantees, the latter better performance guarantees. Which of these is more important will depend on the details of the application.

3.3 The Revelation Principle and the Irrelevance of Truthfulness

The Revelation Principle states that, given requirement (1) in Section 3.1, there is no need to relax requirement (2): it comes "for free."

Theorem 3.1 (Revelation Principle) *For every mechanism M in which every participant has a dominant strategy (no matter what its private information), there is an equivalent direct-revelation DSIC mechanism M'.*

Proof: The proof uses a simulation argument; see Figure 2. By assumption, for every participant i and private information v_i that i might have, i has a dominant strategy $s_i(v_i)$ in the given mechanism M.

Construct the following mechanism M', to which participants delegate the responsibility of playing the appropriate dominant strategy. Precisely, (direct-revelation) mechanism M' accepts sealed bids $b_1, \ldots, b_n$ from the players. It submits the bids $s_1(b_1), \ldots, s_n(b_n)$ to the mechanism M, and chooses the same outcome (e.g., winners of an auction and selling prices) that M does.

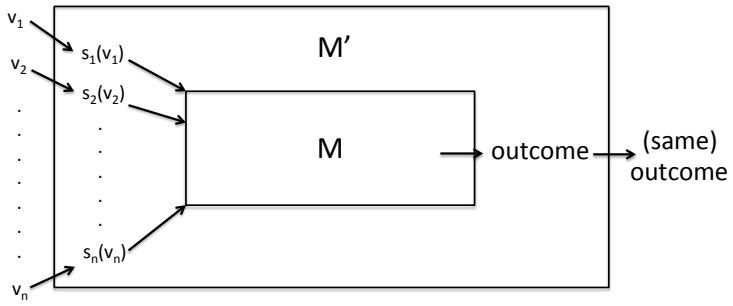

Figure 2: Proof of the Revelation Principle. Construction of the direct-revelation mechanism M', given a mechanism M with dominant strategies.

Mechanism M' is DSIC: If a participant i has private information v_i, then submitting a bid other than v_i can only result in M' playing a strategy other than $s_i(v_i)$ in M, which can only decrease i's utility. ■

The point of Theorem 3.1 is that, at least in principle, if you design a mechanism to have dominant strategies, then you might as well design for direct revelation (in auctions, truthful bidding) to be a dominant strategy.

Many equilibrium concepts other than dominant-strategy equilibria, such as Bayes-Nash equilibria, have their own Revelation Principle. Such principles state that, given the choice of incentive constraints, direct revelation is without loss of generality. Thus, *truthfulness per se is not important*; what makes mechanism design hard is the requirement that a desired outcome (without loss of generality, truthful reporting) in an equilibrium of some type. Varying the choice of equilibrium concept can lead to quite different mechanism design theories, with stronger equilibrium concepts (like dominant-strategy equilibria) requiring weaker behavioral assumptions but with narrower reach than weaker equilibrium concepts (like Bayes-Nash equilibria).

References

[1] S. Chawla, N. Immorlica, and B. Lucier. On the limits of black-box reductions in mechanism design. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 435–448, 2012.