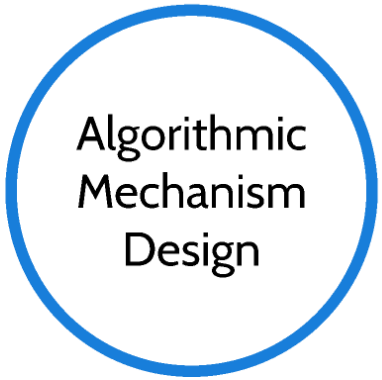
Mechanism Design

hs1@illinois.edu



Ideal auctions

satisfy three key properties:
DSIC; welfare maximizing;
computationally efficient

Mechanisms

Mechanisms and games are different: A mechanism designer sets the **rules of the game**, usually to satisfy an optimization criterion (e.g. social welfare)

A “mechanism” is a **protocol** that interacts with participants and determines a solution to the underlying optimization problem.

first price vs.
second price
auctions

There are two parts to a mechanism: an **allocation** algorithm to select a winner, and a **payment** algorithm to determine payments based on bids

Auctions and Incentive Compatibility

Definition 2.3 (Dominant-Strategy Incentive Compatible) An auction is *dominant-strategy incentive compatible (DSIC)* if truthful bidding is always a dominant strategy for every bidder and if truthful bidders always obtain nonnegative utility.[5]

1 if i wins

$$\sum_{i=1}^{n} v_i x_i,$$

social welfare

Theorem 2.4 (Second-Price Auctions Are Ideal) *A second-price single-item auction satisfies the following:*

(1) ***[strong incentive guarantees]*** *It is a DSIC auction.*

(2) ***[strong performance guarantees]*** *It is welfare maximizing.*

(3) ***[computational efficiency]*** *It can implemented in time polynomial (indeed, linear) in the size of the input, meaning the number of bits necessary to represent the numbers* $v_1, \ldots, v_n$.

all three properties are important

In auctions, there are two design problems: who wins what; who pays what

In auctions, there are two design problems: who wins what; who pays what

1

Assume that bidders bid truthfully; how do we allocate goods so that we can ensure **welfare maximization** and **computational efficiency**?

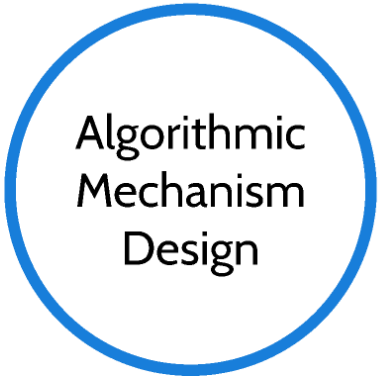
In auctions, there are two design problems: who wins what; who pays what

1 Assume that bidders bid truthfully; how do we allocate goods so that we can ensure **welfare maximization** and **computational efficiency**?

2 how do we set prices so that the auction bids are **incentive compatible**?

Mechanism Design

hs1@illinois.edu



Let an environment have n agents. Each agent i has a private non-negative valuation v_i. A *feasible set* X is a non-negative $n-$vector $(x_1, x_2, \ldots, x_n)$, where x_i denotes the "amount of stuff" given to agent i.

Example 3.1 (Single-Item Auction) In a single-item auction (Section 2.1), X is the set of 0-1 vectors that have at most one 1—that is, $\sum_{i=1}^{n} x_i \leq 1$.

Example 3.2 (k-Unit Auction) With k identical items and the constraint that each bidder gets at most one (Exercise 2.3), the feasible set is the set of 0-1 vectors that satisfy $\sum_{i=1}^{n} x_i \leq k$.

Example 3.3 (Sponsored Search Auction) In a sponsored search auction (Section 2.6), X is the set of n-vectors corresponding to assignments of bidders to slots, where each slot is assigned to at most one bidder and each bidder is assigned to at most one slot. If bidder i is assigned to slot j, then the component x_i equals the click-through rate α_j of her slot.

Example 3.4 (Public Project) Deciding whether or not to build a public project that can be used by all, such as a new bridge, can be modeled by the set $X = \{(0, 0, \ldots, 0), (1, 1, \ldots, 1)\}$.

Recall that a sealed-bid auction has to make two choices: who wins and who pays what. These two decisions are formalized via an **allocation** rule and a **payment** rule, respectively. Here are the three steps:

1. Collect bids $\mathbf{b} = (b_1, \ldots, b_n)$ from all agents. The vector $\mathbf{b}$ is called the *bid vector* or *bid profile*.

2. **[allocation rule]** Choose a feasible allocation $\mathbf{x}(\mathbf{b}) \in X \subseteq \mathbb{R}^n$ as a function of the bids.

3. **[payment rule]** Choose payments $\mathbf{p}(\mathbf{b}) \in \mathbb{R}^n$ as a function of the bids.

Recall that a sealed-bid auction has to make two choices: who wins and who pays what. These two decisions are formalized via an **allocation** rule and a **payment** rule, respectively. Here are the three steps:

1. Collect bids $\mathbf{b} = (b_1, \ldots, b_n)$ from all agents. The vector $\mathbf{b}$ is called the *bid vector* or *bid profile*.

2. **[allocation rule]** Choose a feasible allocation $\mathbf{x}(\mathbf{b}) \in X \subseteq \mathbb{R}^n$ as a function of the bids.

3. **[payment rule]** Choose payments $\mathbf{p}(\mathbf{b}) \in \mathbb{R}^n$ as a function of the bids.

Procedures of this type are called **direct-revelation** mechanisms, because in the first step agents are asked to reveal directly their private valuations.

Agent i receives utility, given allocation and payment rules $\mathbf{x}$ and $\mathbf{p}$:

$$u_i(\mathbf{b}) = v_i \cdot x_i(\mathbf{b}) - p_i(\mathbf{b})$$

Where the payment rules satisfy:

$$p_i(\mathbf{b}) \in [0, b_i \cdot x_i(\mathbf{b})]$$

Definition 3.5 (Implementable Allocation Rule) An allocation rule $\mathbf{x}$ for a single-parameter environment is *implementable* if there is a payment rule $\mathbf{p}$ such that the direct-revelation mechanism $(\mathbf{x}, \mathbf{p})$ is DSIC.

Definition 3.6 (Monotone Allocation Rule) An allocation rule $\mathbf{x}$ for a single-parameter environment is *monotone* if for every agent i and bids $\mathbf{b}_{-i}$ by the other agents, the allocation $x_i(z, \mathbf{b}_{-i})$ to i is nondecreasing in her bid z.

Theorem 3.7 (Myerson's Lemma) *Fix a single-parameter environment.*

(a) An allocation rule $\mathbf{x}$ *is implementable if and only if it is monotone.*

(b) If $\mathbf{x}$ *is monotone, then there is a unique payment rule for which the direct-revelation mechanism* $(\mathbf{x}, \mathbf{p})$ *is DSIC and* $p_i(\mathbf{b}) = 0$ *whenever* $b_i = 0$.

(c) The payment rule in (b) is given by an explicit formula.[2]

Proof sketch:

As shorthand, we will use $x(z)$ and $p(z)$ for the allocation $x_i(z, \mathbf{b}_i)$ and payment $p_i(z, \mathbf{b}_i)$ of i when she bids z, respectively.

A swapping trick

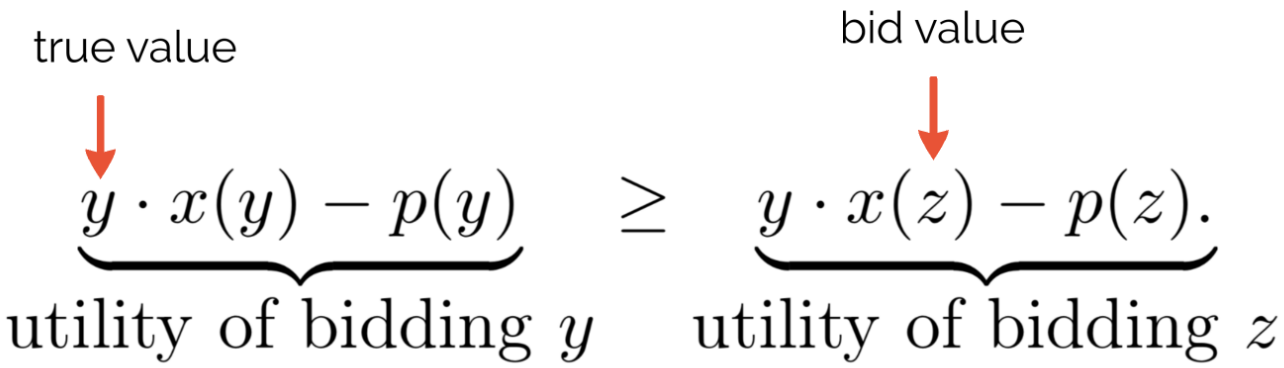
Suppose $(\mathbf{x}, \mathbf{p})$ is DSIC, and consider any $0 \leq y < z$.

true value

bid value

$$\underbrace{z \cdot x(z) - p(z)}_{\text{utility of bidding } z} \geq \underbrace{z \cdot x(y) - p(y).}_{\text{utility of bidding } y}$$

it must also be true that



it must also be true that

true value

bid value

$$\underbrace{y \cdot x(y) - p(y)}_{\text{utility of bidding } y} \geq \underbrace{y \cdot x(z) - p(z).}_{\text{utility of bidding } z}$$

$$z \cdot [x(y) - x(z)] \leq p(y) - p(z) \leq y \cdot [x(y) - x(z)]$$

it must also be true that

true value

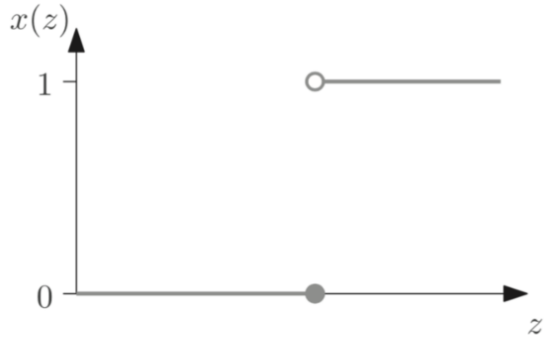
bid value

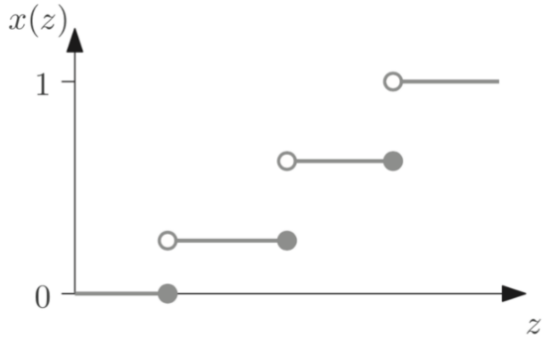
$$\underbrace{y \cdot x(y) - p(y)}_{\text{utility of bidding } y} \geq \underbrace{y \cdot x(z) - p(z).}_{\text{utility of bidding } z}$$

$$z \cdot [x(y) - x(z)] \leq p(y) - p(z) \leq y \cdot [x(y) - x(z)]$$

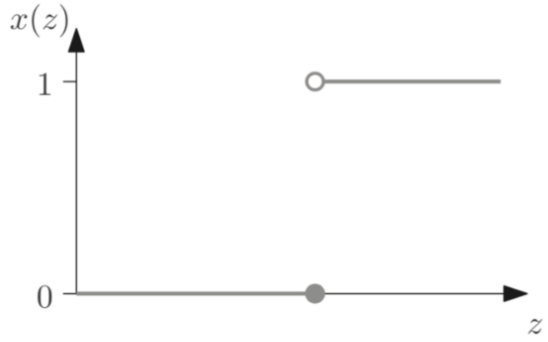
an implementable allocation rule is **monotone**

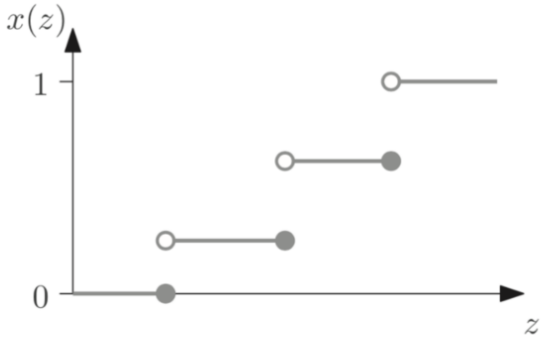
Examples of allocation curves





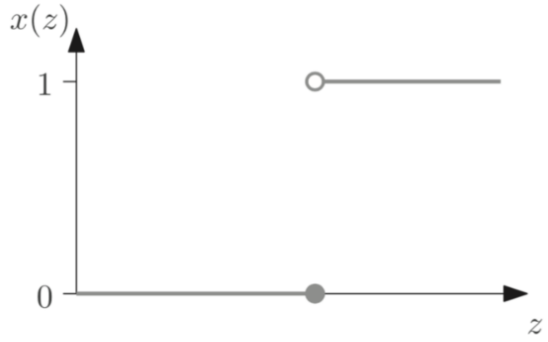
Examples of allocation curves

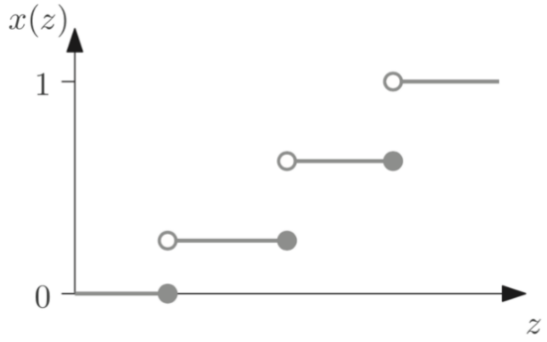




$$z \cdot [x(y) - x(z)] \leq p(y) - p(z) \leq y \cdot [x(y) - x(z)]$$

Examples of allocation curves

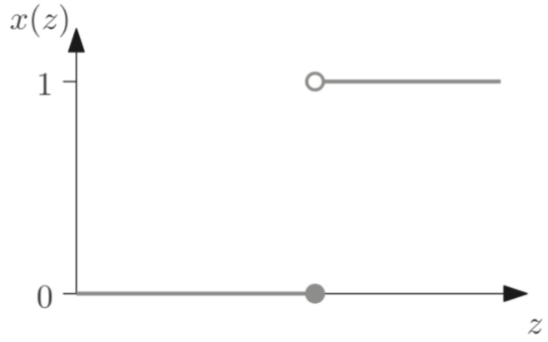


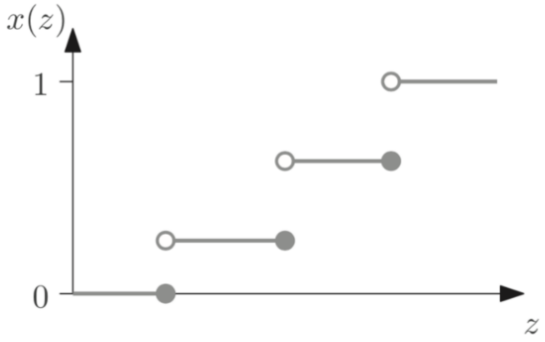


$$z \cdot [x(y) - x(z)] \leq p(y) - p(z) \leq y \cdot [x(y) - x(z)]$$

taking the limit $y \downarrow z$

Examples of allocation curves





$$z \cdot [x(y) - x(z)] \leq p(y) - p(z) \leq y \cdot [x(y) - x(z)]$$

taking the limit $y \downarrow z$

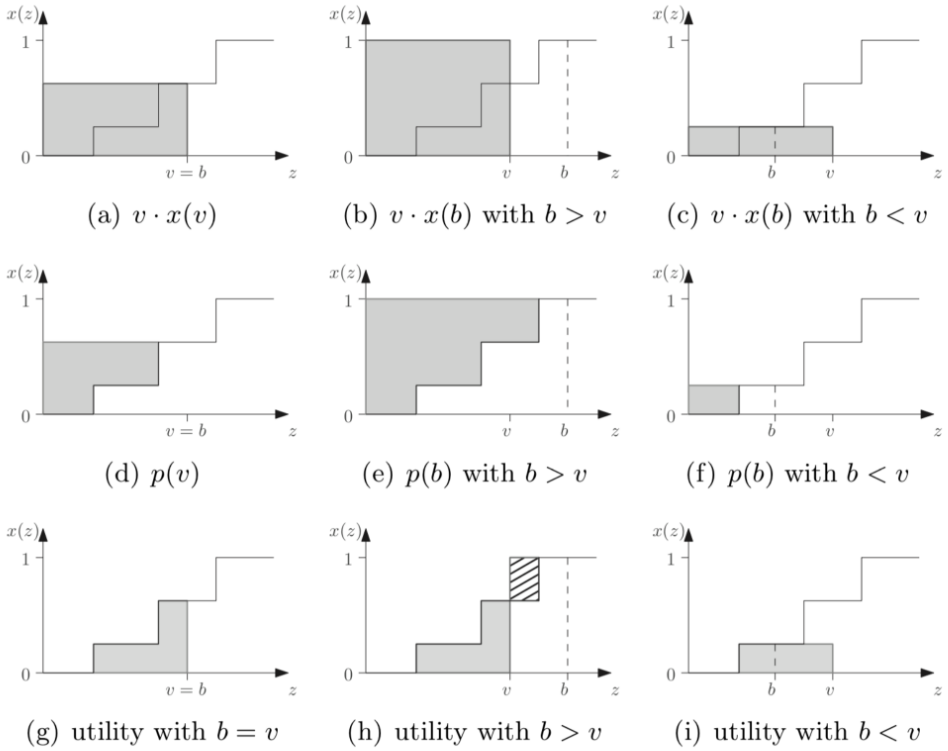
$$\text{jump in } p \text{ at } z = z \cdot [\text{jump in } x \text{ at } z].$$

Combining this with the initial condition $p(0) = 0$, we've derived the following *payment formula*, for every agent i, bids $\mathbf{b}_{-i}$ by other agents, and bid b_i by i:

$$p_i(b_i, \mathbf{b}_{-i}) = \sum_{j=1}^{\ell} z_j \cdot [\text{jump in } x_i(\cdot, \mathbf{b}_{-i}) \text{ at } z_j], \tag{3.5}$$

where $z_1, \ldots, z_\ell$ are the breakpoints of the allocation function $x_i(\cdot, \mathbf{b}_{-i})$ in the range $[0, b_i]$.

When allocation is monotone



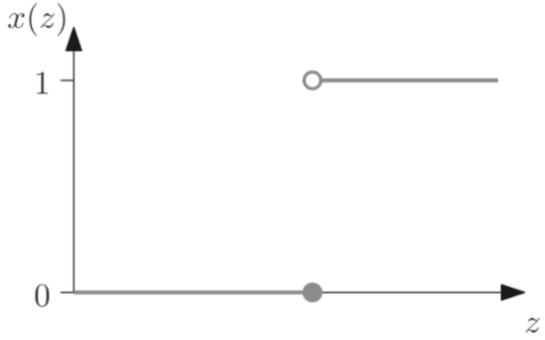
(a) $v \cdot x(v)$ (b) $v \cdot x(b)$ with $b > v$ (c) $v \cdot x(b)$ with $b < v$

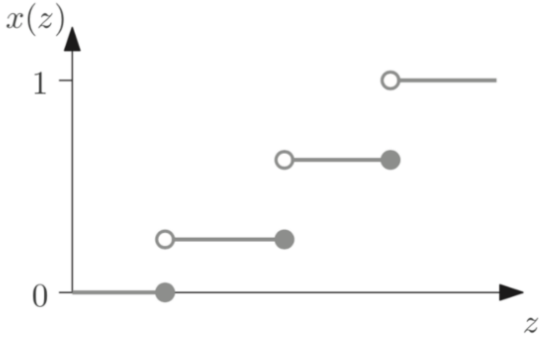
(d) $p(v)$ (e) $p(b)$ with $b > v$ (f) $p(b)$ with $b < v$

(g) utility with $b = v$ (h) utility with $b > v$ (i) utility with $b < v$

In the second priced auction

the key quantity to focus on is: $B = \max_{j \neq i} b_j$





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

key ideas

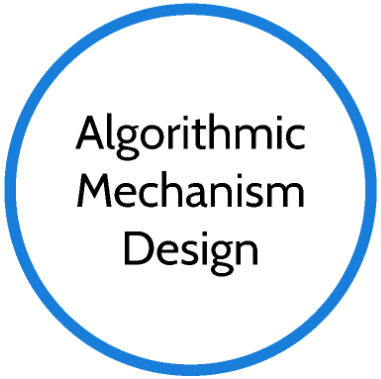
We need an **allocation** rule and a **payment** rule

An allocation rule is **implementable** if there exists a payment rule that extends it to a DSIC mechanism.

Myerson's lemma states that an allocation rule is implementable **if and only if it is monotone**. In this case, the corresponding payment rule is **unique**

Mechanism Design

hs1@illinois.edu



Algorithmic Mechansim Design

Algorithmic mechanism design studies optimization problems where the underlying data—such as the values of goods and costs of performing a task— is initially unknown to the algorithm designer, and must be implicitly or explicitly elicited from self-interested participants.

auctions are a canonical example

Knapsack auctions

Example 4.1 (Knapsack Auction) In a knapsack auction, each bidder i has a publicly known *size* w_i and a private valuation. The seller has a capacity W. The feasible set X is defined as the 0-1 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.)

To maximize social welfare:

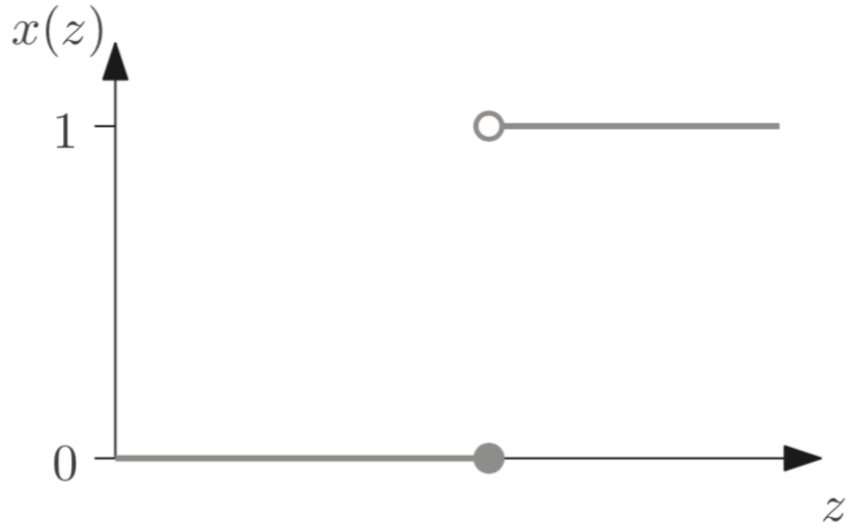
$$\mathbf{x}(\mathbf{b}) = \operatorname{argmax}_X \sum_{i=1}^{n} b_i x_i$$

To maximize social welfare:

$$\mathbf{x}(\mathbf{b}) = \text{argmax}_X \sum_{i=1}^{n} b_i x_i$$

Myerson's lemma (Theorem 3.7, parts (a) and (b)) guarantees the existence of a payment rule p such that the mechanism $(\mathbf{x}, \mathbf{p})$ is DSIC.

Since the allocation rule is monotone and assigns 0 or 1 to every bidder, the allocation curve $x_i(\cdot, \mathbf{b}_i)$ is 0 until some breakpoint z, at which point it jumps to 1



Is this an ideal mechanism?

that is, it:

(1) is DSIC;

(2) is welfare maximizing, assuming truthful bids; and

(3) runs in time polynomial in the input size, which is the number of bits needed to represent all of the relevant numbers (bids, sizes, and the capacity).[2]

The answer is No!

The reason is that the knapsack problem is $\mathcal{NP}$-hard. This means that there is no polynomial-time implementation of the allocation rule unless $\mathcal{P} = \mathcal{NP}$. Thus, properties (2) and (3) are incompatible.

The dominant paradigm in algorithmic mechanism design is to relax the second requirement of ideal auctions (welfare maximization) as little as possible, subject to the first (DSIC) and third (polynomial-time) requirements.

Can we get DSIC for free?

One reason there has been so much progress in algorithmic mechanism design over the past 15 years is its strong resemblance to the mature field of approximation algorithms.

The primary goal in approximation algorithms is to design polynomial-time algorithms for NP-hard optimization problems that are as close to optimal as possible. Algorithmic mechanism design has exactly the same goal, except that the algorithms must additionally obey a monotonicity constraint.

Exact welfare maximization automatically yields a monotone allocation rule; Does an analogous fact hold for approximate welfare maximization?

A Greedy Knapsack Heuristic

1. Sort and re-index the bidders so that

$$\frac{b_1}{w_1} \geq \frac{b_2}{w_2} \geq \cdots \geq \frac{b_n}{w_n}.$$ [6]

2. Pick winners in this order until one doesn't fit, and then halt.[7]

3. Return either the solution from the previous step or the highest bidder, whichever has larger social welfare.[8]

this is a 1/2 approximation solution

While this rule is monotone, this is not always true; we need to check and possibly tweak the algorithm to ensure monotonicity

The revelation principle

Theorem 4.3 (Revelation Principle for DSIC Mechanisms) *For every mechanism M in which every participant always has a dominant strategy, there is an equivalent direct-revelation DSIC mechanism M'.*

Mechanism Design

hs1@illinois.edu