Machine Learning & Mechanism Design:

Dynamic and Discriminatory Pricing in Auctions

Jason D. Hartline

Microsoft Research – Silicon Valley

(Joint with with Maria-Florina Balcan,

Avrim Blum, and Yishay Mansour)

August 5, 2005

The Problem

Sellers can extract more of surplus with discriminatory pricing.

The Problem

Sellers can extract more of surplus with discriminatory pricing.

Two approaches:

1. Distinguish between products.
   (E.g., software versioning, airline tickets, etc.)

2. Price discriminate with observable customer features.
   (E.g., college tuition, DVDs, car insurance, shipping)

The Problem

Sellers can extract more of surplus with discriminatory pricing.

Two approaches:

1. Distinguish between products.
   (E.g., software versioning, airline tickets, etc.)

2. Price discriminate with observable customer features.
   (E.g., college tuition, DVDs, car insurance, shipping)

Goal: design mechanism to optimally price discriminate.

Optimal Mechanism Design

Typical Economic approach to optimal mechanism design:

- Assume valuations are from known distribution.
- Design optimal auction for distribution.

Optimal Mechanism Design

Typical Economic approach to optimal mechanism design:

- Assume valuations are from known distribution.
- Design optimal auction for distribution.

Notes on optimal mechanism design problem:

- Solved by Myerson (for single-parameter case).
- non-identical distributions $\Longrightarrow$ discriminatory pricing.

Optimal Mechanism Design

Typical Economic approach to optimal mechanism design:

- Assume valuations are from known distribution.
- Design optimal auction for distribution.

Notes on optimal mechanism design problem:

- Solved by Myerson (for single-parameter case).
- non-identical distributions $\Longrightarrow$ discriminatory pricing.
- Assumed known distribution ignores:
  - incentives (of acquiring distribution)
  - performance (from inaccurate distribution)

Optimal Mechanism Design

Typical Economic approach to optimal mechanism design:

- Assume valuations are from known distribution.
- Design optimal auction for distribution.

Notes on optimal mechanism design problem:

- Solved by Myerson (for single-parameter case).
- non-identical distributions $\Longrightarrow$ discriminatory pricing.
- Assumed known distribution ignores:
  - incentives (of acquiring distribution)
  - performance (from inaccurate distribution)

Goal: understand how quality and incentives of learning distribution affect profit.

Setting

1. *Unlimited supply* of stuff to sell.

2. bidders with private *valuations* for stuff.

3. make each bidder an *offer*.

4. revenue is *incentive compatible* function of offer and valuation.

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))

2. bidders with private *valuations* for stuff.

3. make each bidder an *offer*.

4. revenue is *incentive compatible* function of offer and valuation.

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: "PV worth \$400, SV worth \$300")

3. make each bidder an *offer*.

4. revenue is *incentive compatible* function of offer and valuation.

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: “PV worth $400, SV worth $300”)

3. make each bidder an *offer*.
   (Example 1: Seller: “PV costs $369.88, SV costs $124.99”)

4. revenue is *incentive compatible* function of offer and valuation.

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: “PV worth $400, SV worth $300”)

3. make each bidder an *offer*.
   (Example 1: Seller: “PV costs $369.88, SV costs $124.99”)

4. revenue is *incentive compatible* function of offer and valuation.
   (Example 1: Sold: SV for $124.99!)

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))
   (Example 2: Tuition for in state (IS) and out of state (OS) students)

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: “PV worth $400, SV worth $300”)

3. make each bidder an *offer*.
   (Example 1: Seller: “PV costs $369.88, SV costs $124.99”)

4. revenue is *incentive compatible* function of offer and valuation.
   (Example 1: Sold: SV for $124.99!)

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))
   (Example 2: Tuition for in state (IS) and out of state (OS) students)

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: “PV worth $400, SV worth $300”)
   (Example 2: Bidder (OS): “Tuition worth $15,000”)

3. make each bidder an *offer*.
   (Example 1: Seller: “PV costs $369.88, SV costs $124.99”)

4. revenue is *incentive compatible* function of offer and valuation.
   (Example 1: Sold: SV for $124.99!)

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))
   (Example 2: Tuition for in state (IS) and out of state (OS) students)

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: "PV worth $400, SV worth $300")
   (Example 2: Bidder (OS): "Tuition worth $15,000")

3. make each bidder an *offer*.
   (Example 1: Seller: "PV costs $369.88, SV costs $124.99")
   (Example 2: Seller: "IS costs $9,256.80, OS costs $16,855.30,")

4. revenue is *incentive compatible* function of offer and valuation.
   (Example 1: Sold: SV for $124.99!)

Setting

1. *Unlimited supply* of stuff to sell.
   (Example 1: MS Office Professional (PV) & Student Version (SV))
   (Example 2: Tuition for in state (IS) and out of state (OS) students)

2. bidders with private *valuations* for stuff.
   (Example 1: Bidder: "PV worth $400, SV worth $300")
   (Example 2: Bidder (OS): "Tuition worth $15,000")

3. make each bidder an *offer*.
   (Example 1: Seller: "PV costs $369.88, SV costs $124.99")
   (Example 2: Seller: "IS costs $9,256.80, OS costs $16,855.30,")

4. revenue is *incentive compatible* function of offer and valuation.
   (Example 1: Sold: SV for $124.99!)
   (Example 2: No Sale!)

Overview

⟹ 1. Auction Problem

   (a) Random Sampling Solution

   (b) Retrospective bounds.

   (c) Software Versioning Example.

2. Online Auction Problem

   (a) Expert Learning based Auction.

   (b) Expert Learning with non-uniform bounds.

3. Conclusions

Auction Problem

The *Unlimited Supply Auction Problem*:

Given:

- unlimited supply of stuff.
- Set S of n bidders with valuations for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Auction with profit near that of optimal single offer.

Auction Problem

The *Unlimited Supply Auction Problem*:

Given:

- unlimited supply of stuff.
- Set S of n bidders with valuations for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Auction with profit near that of optimal single offer.

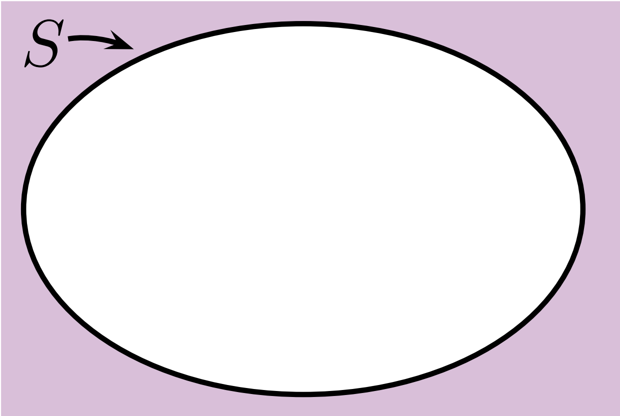
Notation:

- $g(i) =$ payoff from bidder i when offered g.
- $g(S) = \sum_{i \in S} g(i)$.
- $\mathrm{opt}_{\mathcal{G}}(S) = \mathrm{argmax}_{g \in \mathcal{G}}\, g(S)$.
- $\mathrm{OPT}_{\mathcal{G}}(S) = \max_{g \in \mathcal{G}}\, g(S)$.

Random Sampling Auction

Random Sampling Optimal Offer Auction, $\text{RSOO}_{\mathcal{G}}$

1. Randomly partition bidders into two sets: S_1 and S_2.
2. compute g_1 (resp. g_2), optimal offer for S_1 (resp. S_2)
3. Offer g_1 to S_2 and g_2 to S_1.

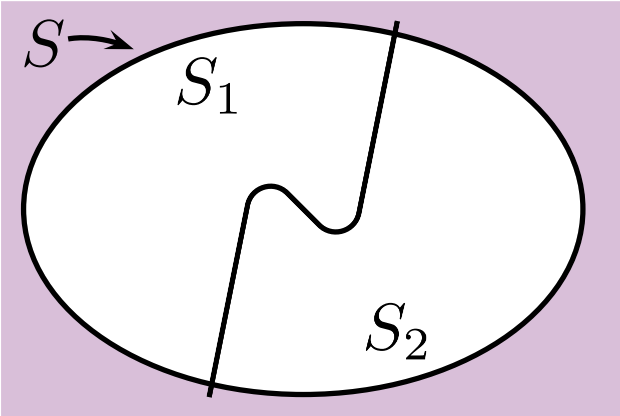


(Random Sampling Auction from [Goldberg, Hartline, Wright 2001])

Random Sampling Auction

Random Sampling Optimal Offer Auction, $RSOO_{\mathcal{G}}$

1. Randomly partition bidders into two sets: S_1 and S_2.
2. compute g_1 (resp. g_2), optimal offer for S_1 (resp. S_2)
3. Offer g_1 to S_2 and g_2 to S_1.

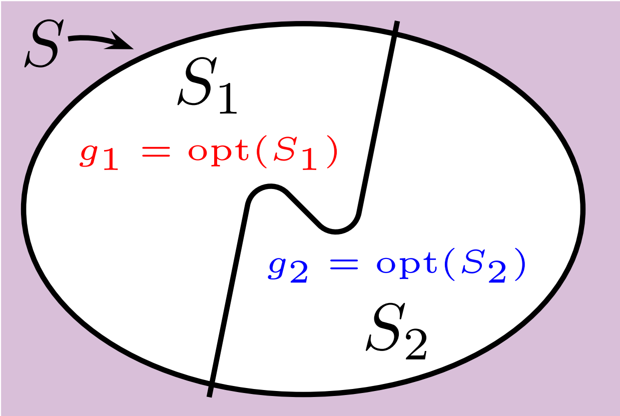


(Random Sampling Auction from [Goldberg, Hartline, Wright 2001])

Random Sampling Auction

Random Sampling Optimal Offer Auction, $\text{RSOO}_{\mathcal{G}}$

1. Randomly partition bidders into two sets: S_1 and S_2.
2. compute g_1 (resp. g_2), optimal offer for S_1 (resp. S_2)
3. Offer g_1 to S_2 and g_2 to S_1.

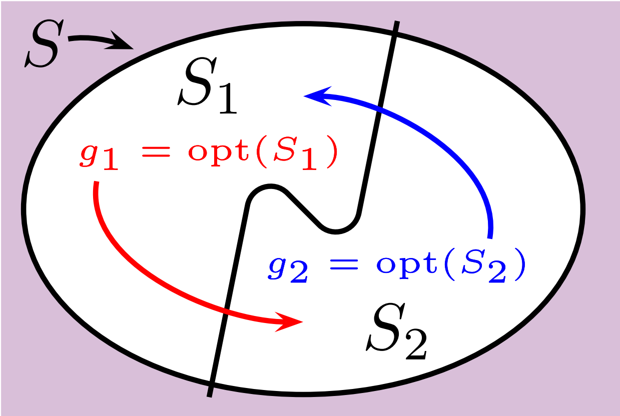


(Random Sampling Auction from [Goldberg, Hartline, Wright 2001])

Random Sampling Auction

Random Sampling Optimal Offer Auction, $\mathrm{RSOO}_{\mathcal{G}}$

1. Randomly partition bidders into two sets: S_1 and S_2.
2. compute g_1 (resp. g_2), optimal offer for S_1 (resp. S_2)
3. Offer g_1 to S_2 and g_2 to S_1.

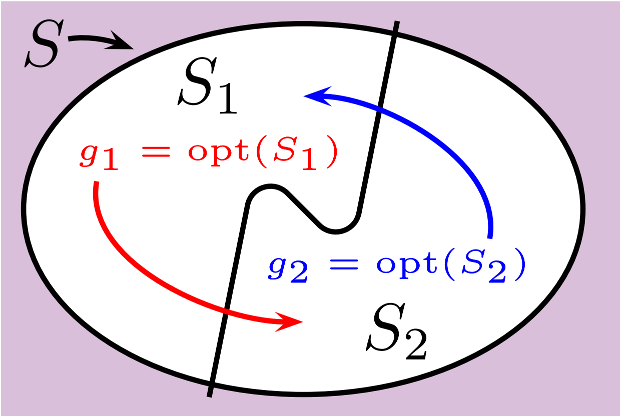


(Random Sampling Auction from [Goldberg, Hartline, Wright 2001])

Random Sampling Auction

Random Sampling Optimal Offer Auction, RSOO$_{\mathcal{G}}$

1. Randomly partition bidders into two sets: S_1 and S_2.
2. compute g_1 (resp. g_2), optimal offer for S_1 (resp. S_2)
3. Offer g_1 to S_2 and g_2 to S_1.



Question: when is RSOO$_{\mathcal{G}}$ good?

(Random Sampling Auction from [Goldberg, Hartline, Wright 2001])

Performance Analysis

Lemma: For g and random partitions S_1 and S_2:

$$\textbf{Pr}[|g(S_1) - g(S_2)| > \epsilon \max(p, g(S))] \leq 2e^{-\epsilon^2 p/2h}.$$

Performance Analysis

Lemma: For g and random partitions S_1 and S_2:

$$\mathbf{Pr}[|g(S_1) - g(S_2)| > \epsilon \max(p, g(S))] \leq 2e^{-\epsilon^2 p/2h}.$$

Consider:

- Use $p = \mathrm{OPT}_{\mathcal{G}}$.
- If $|\mathcal{G}|\, e^{-\epsilon^2 \,\mathrm{OPT}_{\mathcal{G}}\,/2h} \leq \delta$,
- union bound probability any $g \in \mathcal{G}$ is bad by δ.

Performance Analysis

Lemma: For g and random partitions S_1 and S_2:

$$\mathbf{Pr}[|g(S_1) - g(S_2)| > \epsilon \max(p, g(S))] \leq 2e^{-\epsilon^2 p/2h}.$$

Consider:

- Use $p = \mathrm{OPT}_{\mathcal{G}}$.
- If $|\mathcal{G}|\, e^{-\epsilon^2 \mathrm{OPT}_{\mathcal{G}}/2h} \leq \delta$,
- union bound probability any $g \in \mathcal{G}$ is bad by δ.

Theorem: With probability $1 - \delta$ profit from RSOO$_{\mathcal{G}}$ is at least

$$(1 - \epsilon)\,\mathrm{OPT}_{\mathcal{G}} - O(\tfrac{h}{\epsilon^2} \log \tfrac{|\mathcal{G}|}{\delta})$$

Performance Analysis

Lemma: For g and random partitions S_1 and S_2:

$$\mathbf{Pr}[|g(S_1) - g(S_2)| > \epsilon \max(p, g(S))] \leq 2e^{-\epsilon^2 p/2h}.$$

Consider:

- Use $p = \mathrm{OPT}_{\mathcal{G}}$.
- If $|\mathcal{G}|\, e^{-\epsilon^2 \mathrm{OPT}_{\mathcal{G}} / 2h} \leq \delta$,
- union bound probability any $g \in \mathcal{G}$ is bad by δ.

Theorem: With probability $1 - \delta$ profit from RSOO$_{\mathcal{G}}$ is at least

$$(1 - \epsilon)\,\mathrm{OPT}_{\mathcal{G}} - O(\tfrac{h}{\epsilon^2} \log \tfrac{|\mathcal{G}|}{\delta})$$

Interpretation: $O(h \log |\mathcal{G}|)$ is *convergence time*.

Example

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{$price 2^i for $i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $O(h \log |\mathcal{G}|) = O(h \log \log h)$

Overview

1. Auction Problem
   (a) Random Sampling Solution
   ⟹ (b) Retrospective bounds.
   (c) Software Versioning Example.
2. Online Auction Problem
   (a) Expert Learning based Auction.
   (b) Expert Learning with non-uniform bounds.
3. Conclusions

$|\mathcal{G}| = \infty$?

What if $|\mathcal{G}| = \infty$?

$|\mathcal{G}| = \infty$?

What if $|\mathcal{G}| = \infty$?

Example: Selling tee shirts. (non-discretized prices)

- Bidders with valuations $v_1, \ldots, v_n$ in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{\text{price } p \in [1, h]\}$.
- Convergence Time:

$|\mathcal{G}| = \infty$?

What if $|\mathcal{G}| = \infty$?

Example: Selling tee shirts. (non-discretized prices)

- Bidders with valuations $v_1, \ldots, v_n$ in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{$price $p \in [1, h]\}$.
- Convergence Time:

Observation:

- Suppose RSOO$_{\mathcal{G}}$ on S only offers $g \in \mathcal{G}_S \subset \mathcal{G}$.
- Then RSOO$_{\mathcal{G}_S}(S)$ is same as RSOO$_{\mathcal{G}}(S)$.
- Retrospectively perform analysis on $\mathcal{G}_S$ instead of $\mathcal{G}$.

What if $|\mathcal{G}| = \infty$?

Example: Selling tee shirts. (non-discretized prices)

- Bidders with valuations $v_1, \ldots, v_n$ in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{\text{price } p \in [1, h]\}$.
- Convergence Time:

Observation:

- Suppose $\text{RSOO}_{\mathcal{G}}$ on S only offers $g \in \mathcal{G}_S \subset \mathcal{G}$.
- Then $\text{RSOO}_{\mathcal{G}_S}(S)$ is same as $\text{RSOO}_{\mathcal{G}}(S)$.
- Retrospectively perform analysis on $\mathcal{G}_S$ instead of $\mathcal{G}$.

Consider: $\mathcal{G}_S = \{\text{“offer } v_i\text{”} \ : \ i \in S\}$.

$|\mathcal{G}| = \infty$?

What if $|\mathcal{G}| = \infty$?

Example: Selling tee shirts. (non-discretized prices)

- Bidders with valuations $v_1, \ldots, v_n$ in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{\text{price } p \in [1, h]\}$.
- Convergence Time: $O(h\,|\mathcal{G}_S|) = O(h \log n)$

Observation:

- Suppose $\text{RSOO}_{\mathcal{G}}$ on S only offers $g \in \mathcal{G}_S \subset \mathcal{G}$.
- Then $\text{RSOO}_{\mathcal{G}_S}(S)$ is same as $\text{RSOO}_{\mathcal{G}}(S)$.
- Retrospectively perform analysis on $\mathcal{G}_S$ instead of $\mathcal{G}$.

Consider: $\mathcal{G}_S = \{\text{“offer } v_i\text{”} \;:\; i \in S\}$.

Example: Software Versioning

Example: Software versioning.

- Microsoft has M possible versions of MS Office,
- but can only package and sell $m \ll M$ distinct versions. (E.g., Student and Professional).

Example: Software Versioning

Example: Software versioning.

- Microsoft has M possible versions of MS Office,
- but can only package and sell $m \ll M$ distinct versions. (E.g., Student and Professional).
- Reasonable offers: $\mathcal{G} = \{\mathbf{p} \in (\mathbb{R} \cup \{\infty\})^M$ with $p_j = \infty$ for all but m items$\}$.

Example: Software Versioning

Example: Software versioning.

- Microsoft has M possible versions of MS Office,
- but can only package and sell $m \ll M$ distinct versions. (E.g., Student and Professional).
- Reasonable offers: $\mathcal{G} = \{\mathbf{p} \in (\mathbb{R} \cup \{\infty\})^M$ with $p_j = \infty$ for all but m items$\}$.
- What is $|\mathcal{G}_S|$?

Example: Software Versioning

Example: Software versioning.

- Microsoft has M possible versions of MS Office,
- but can only package and sell $m \ll M$ distinct versions. (E.g., Student and Professional).
- Reasonable offers: $\mathcal{G} = \{\mathbf{p} \in (\mathbb{R} \cup \{\infty\})^M$ with $p_j = \infty$ for all but m items$\}$.
- What is $|\mathcal{G}_S|$?

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Example: Software Versioning

Example: Software versioning.

- Microsoft has M possible versions of MS Office,
- but can only package and sell $m \ll M$ distinct versions. (E.g., Student and Professional).
- Reasonable offers: $\mathcal{G} = \{\mathbf{p} \in (\mathbb{R} \cup \{\infty\})^M$ with $p_j = \infty$ for all but m items$\}$.
- What is $|\mathcal{G}_S|$?

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Convergence time $= O(hm\log(nm^2M))$.

Example: Software Versioning

Example: Software versioning.

- Microsoft has M possible versions of MS Office,
- but can only package and sell $m \ll M$ distinct versions. (E.g., Student and Professional).
- Reasonable offers: $\mathcal{G} = \{\mathbf{p} \in (\mathbb{R} \cup \{\infty\})^M$ with $p_j = \infty$ for all but m items$\}$.
- What is $|\mathcal{G}_S|$?

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Convergence time $= O(hm \log(nm^2M)) \approx O(hm \log n)$.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof:

1. Fix set of m items to sell.

2. Bidder i's valuation divides price space into $m+1$ convex regions.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof:

1. Fix set of m items to sell.
2. Bidder i's valuation divides price space into $m+1$ convex regions.
3. Regions are joined by $(m+1)^2$ hyperplanes.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof:

1. Fix set of m items to sell.
2. Bidder i's valuation divides price space into $m+1$ convex regions.
3. Regions are joined by $(m+1)^2$ hyperplanes.
4. n bidders total for $n(m+1)^2$ hyperplanes.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof:

1. Fix set of m items to sell.
2. Bidder i's valuation divides price space into $m+1$ convex regions.
3. Regions are joined by $(m+1)^2$ hyperplanes.
4. n bidders total for $n(m+1)^2$ hyperplanes.
5. RSOO$_{\mathcal{G}}$ offer price must be at intersection of hyperplanes.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof:

1. Fix set of m items to sell.
2. Bidder i's valuation divides price space into $m + 1$ convex regions.
3. Regions are joined by $(m + 1)^2$ hyperplanes.
4. n bidders total for $n(m + 1)^2$ hyperplanes.
5. RSOO$_{\mathcal{G}}$ offer price must be at intersection of hyperplanes.
6. $K = n(m + 1)^2$ hyperplanes in m dimensions intersect in K^m.

Proof of Lemma

Lemma: $|\mathcal{G}_S| = O((nm^2M)^m)$.

Proof:

1. Fix set of m items to sell.
2. Bidder i's valuation divides price space into $m+1$ convex regions.
3. Regions are joined by $(m+1)^2$ hyperplanes.
4. n bidders total for $n(m+1)^2$ hyperplanes.
5. RSOO$_{\mathcal{G}}$ offer price must be at intersection of hyperplanes.
6. $K = n(m+1)^2$ hyperplanes in m dimensions intersect in K^m.
7. Sum over M^m possible m-item sets.

Other Results

See paper for details on:

- Bounds for $\text{RSOO}_{\mathcal{G}}$ for item-pricing in combinatorial auctions.
- Bounds for $\text{RSOO}_{\mathcal{G}}$ on bidders with observable features.
- Better bounds with *ϵ-covers* of $\mathcal{G}$.
- Better random sampling auction with *structural risk minimization*.
- Using approximation algorithms in $\text{RSOO}_{\mathcal{G}}$.

Overview

1. Auction Problem
   (a) Random Sampling Solution
   (b) Retrospective bounds.
   (c) Software Versioning Example.

⟹ 2. Online Auction Problem
   (a) Expert Learning based Auction.
   (b) Expert Learning with non-uniform bounds.

3. Conclusions

Online Auction Problem

Online Auction Problem:

- unlimited supply of stuff.
- class $\mathcal{G}$ of reasonable offers.
- Bidders arrive one at a time and place bids, $b_1, b_2, \ldots$
- Auctioneer makes offer g from $\mathcal{G}$ before next bidder arrives.
- **Goal:** Auction with profit close to optimal single offer.

Online Auction Problem

Online Auction Problem:

- unlimited supply of stuff.
- class $\mathcal{G}$ of reasonable offers.
- Bidders arrive one at a time and place bids, $b_1, b_2, \ldots$
- Auctioneer makes offer g from $\mathcal{G}$ before next bidder arrives.
- **Goal:** Auction with profit close to optimal single offer.

Two Difficulties:

1. Incentive Compatibility requirement:

2. Online Requirement (do not know future):

Online Auction Problem

Online Auction Problem:

- unlimited supply of stuff.
- class $\mathcal{G}$ of reasonable offers.
- Bidders arrive one at a time and place bids, $b_1, b_2, \ldots$
- Auctioneer makes offer g from $\mathcal{G}$ before next bidder arrives.
- **Goal:** Auction with profit close to optimal single offer.

Two Difficulties:

1. Incentive Compatibility requirement:
   offer to bidder i not function of b_i.
2. Online Requirement (do not know future):

Online Auction Problem

Online Auction Problem:

- unlimited supply of stuff.
- class $\mathcal{G}$ of reasonable offers.
- Bidders arrive one at a time and place bids, $b_1, b_2, \ldots$
- Auctioneer makes offer g from $\mathcal{G}$ before next bidder arrives.
- **Goal:** Auction with profit close to optimal single offer.

Two Difficulties:

1. Incentive Compatibility requirement:
   offer to bidder i not function of b_i.
2. Online Requirement (do not know future):
   price offered bidder i not function of future bids.

Online Auction Problem

Online Auction Problem:

- unlimited supply of stuff.
- class $\mathcal{G}$ of reasonable offers.
- Bidders arrive one at a time and place bids, $b_1, b_2, \ldots$
- Auctioneer makes offer g from $\mathcal{G}$ before next bidder arrives.
- **Goal:** Auction with profit close to optimal single offer.

Two Difficulties:

1. Incentive Compatibility requirement:
   offer to bidder i not function of b_i.
2. Online Requirement (do not know future):
   price offered bidder i not function of future bids.

Conclusion: offer for bidder i based only on prior bids: $b_1, \ldots, b_{i-1}$.

Assumptions

1. We learn each bidders full valuation.

Assumptions

1. We learn each bidders full valuation.

   for partial information case see multi-armed bandit solutions: [Blum, Kumar, Rudra, Wu ’03][Kleinberg, Leighton ’03][Blum, Hartline 05]

Assumptions

1. We learn each bidders full valuation.

   for partial information case see multi-armed bandit solutions:
   [Blum, Kumar, Rudra, Wu ’03][Kleinberg, Leighton ’03][Blum, Hartline 05]

2. Bidders cannot come back.

3. Bidders cannot lie about their arrival time.

Assumptions

1. We learn each bidders full valuation.

   for partial information case see multi-armed bandit solutions:
   [Blum, Kumar, Rudra, Wu ’03][Kleinberg, Leighton ’03][Blum, Hartline 05]

2. Bidders cannot come back.

3. Bidders cannot lie about their arrival time.

   for *temporal strategyproofness* see: [Hajiaghayi, Kleinberg, Parkes ’04]

Assumptions

1. We learn each bidders full valuation.

   for partial information case see multi-armed bandit solutions:
   [Blum, Kumar, Rudra, Wu ’03][Kleinberg, Leighton ’03][Blum, Hartline 05]

2. Bidders cannot come back.

3. Bidders cannot lie about their arrival time.

   for *temporal strategyproofness* see: [Hajiaghayi, Kleinberg, Parkes ’04]

4. items in unlimited supply.

Assumptions

1. We learn each bidders full valuation.

   for partial information case see multi-armed bandit solutions:
   [Blum, Kumar, Rudra, Wu ’03][Kleinberg, Leighton ’03][Blum, Hartline 05]

2. Bidders cannot come back.

3. Bidders cannot lie about their arrival time.

   for *temporal strategyproofness* see: [Hajiaghayi, Kleinberg, Parkes ’04]

4. items in unlimited supply.

   for *limited supply* see:
   [Hajiaghayi, Kleinberg, Parkes ’04][Kleinberg ’05]

Online Learning

Expert Online Learning Problem:

In round i:

1. Each of k experts propose a strategy.
2. We choose an expert's strategy.
3. Find out how each strategy performed (payoff)

Goal: Obtain payoff close to single best expert overall (in hindsight).

Online Learning

Expert Online Learning Problem:

In round i:

1. Each of k experts propose a strategy.
2. We choose an expert's strategy.
3. Find out how each strategy performed (payoff)

Goal: Obtain payoff close to single best expert overall (in hindsight).

Weighted Majority Algorithm: (for round i)

Let h be maximum payoff. For expert j, let s_j be total payoff thus far.

Choose expert j's strategy with probability proportional to $(1+2\epsilon)^{s_j/h}$.

Expert Online Learning Problem:

In round i:

1. Each of k experts propose a strategy.
2. We choose an expert's strategy.
3. Find out how each strategy performed (payoff)

Goal: Obtain payoff close to single best expert overall (in hindsight).

Weighted Majority Algorithm: (for round i)

Let h be maximum payoff. For expert j, let s_j be total payoff thus far.

Choose expert j's strategy with probability proportional to $(1+2\epsilon)^{s_j/h}$.

Result: $\mathbf{E}[\text{payoff}] = (1-\epsilon)\,\mathrm{OPT} - \frac{h}{2\epsilon}\log k$.

Application to Online Auctions

Application: (to online auctions) [Blum Kumar Rudra Wu 2003]

Application to Online Auctions

Application: (to online auctions) [Blum Kumar Rudra Wu 2003]

1. Expert for each $g \in \mathcal{G}$

Application to Online Auctions

Application: (to online auctions) [Blum Kumar Rudra Wu 2003]

1. Expert for each $g \in \mathcal{G}$
2. Best expert $\Rightarrow$ best offer.

Application to Online Auctions

Application: (to online auctions) [Blum Kumar Rudra Wu 2003]

1. Expert for each $g \in \mathcal{G}$
2. Best expert $\Rightarrow$ best offer.

Result: $\mathbf{E}[\text{profit}] = (1-\epsilon)\,\mathrm{OPT}_{\mathcal{G}} - \frac{h}{\epsilon}\log|\mathcal{G}|$.

Application to Online Auctions

Application: (to online auctions) [Blum Kumar Rudra Wu 2003]

1. Expert for each $g \in \mathcal{G}$
2. Best expert $\Rightarrow$ best offer.

Result: $\mathbf{E}[\text{profit}] = (1-\epsilon)\,\mathrm{OPT}_{\mathcal{G}} - \frac{h}{\epsilon}\log|\mathcal{G}|$.

Note: Same convergence time as for $\mathrm{RSOO}_{\mathcal{G}}$.

Example

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{\text{price } 2^i \text{ for } i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $O(h \log |\mathcal{G}|)$

Example

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{\text{price } 2^i \text{ for } i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $O(h \log |\mathcal{G}|) = O(h \log \log h)$.

Better Bounds?

Can we get better bounds?

Retrospective technique like using $\mathcal{G}_S$ does not work.

Overview

1. Auction Problem
   (a) Random Sampling Solution
   (b) Retrospective bounds.
   (c) Software Versioning Example.
2. Online Auction Problem
   (a) Expert Learning based Auction.
   $\Longrightarrow$ (b) Expert Learning with non-uniform bounds.
3. Conclusions

Non-uniform Bounds on Payoff

Expert Online Learning Problem: In round i:

1. Each of k experts propose a strategy.
2. We choose an expert's strategy.
3. Find out how each strategy performed (payoff)
4. Expert i's payoff is always less than h_i.

Goal: Obtain payoff close to single best expert overall (in hindsight).

Non-uniform Bounds on Payoff

Expert Online Learning Problem: In round i:

1. Each of k experts propose a strategy.
2. We choose an expert's strategy.
3. Find out how each strategy performed (payoff)
4. Expert i's payoff is always less than h_i.

Goal: Obtain payoff close to single best expert overall (in hindsight).

Non-uniform Experts Algorithm: [Kalai '01][Blum, Hartline '05]

1. (initialization) For each expert, j, add initial score, s_j, as:

   $$h_i \times \text{number of heads flipped in a row.}$$

2. Run deterministic "go with best expert" algorithm.

Non-uniform Bounds on Payoff

Expert Online Learning Problem: In round i:

1. Each of k experts propose a strategy.
2. We choose an expert's strategy.
3. Find out how each strategy performed (payoff)
4. Expert i's payoff is always less than h_i.

Goal: Obtain payoff close to single best expert overall (in hindsight).

Non-uniform Experts Algorithm: [Kalai '01][Blum, Hartline '05]

1. (initialization) For each expert, j, add initial score, s_j, as:

$$h_i \times \text{number of heads flipped in a row.}$$

2. Run deterministic "go with best expert" algorithm.

Result: $\mathbf{E}[\text{profit}] \geq \text{OPT}/2 - \sum_i h_i$.

Application to Online Auctions

Application: (to online auctions)

Application to Online Auctions

Application: (to online auctions)

1. Bound h_g for each $g \in \mathcal{G}$.
2. Expert for each $g \in \mathcal{G}$
3. Best expert $\Rightarrow$ best offer.

Application to Online Auctions

Application: (to online auctions)

1. Bound h_g for each $g \in \mathcal{G}$.
2. Expert for each $g \in \mathcal{G}$
3. Best expert $\Rightarrow$ best offer.

Result: $\mathbf{E}[\text{profit}] = \mathrm{OPT}_{\mathcal{G}}/2 - \sum_{g \in \mathcal{G}} h_g$.

Application to Online Auctions

Application: (to online auctions)

1. Bound h_g for each $g \in \mathcal{G}$.
2. Expert for each $g \in \mathcal{G}$
3. Best expert $\Rightarrow$ best offer.

Result: $\mathbf{E}[\text{profit}] = \text{OPT}_{\mathcal{G}}/2 - \sum_{g \in \mathcal{G}} h_g$.

Note: Convergence time $= \sum_{g \in \mathcal{G}} h_g$

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{$price 2^i for $i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $\sum_{g \in \mathcal{G}} h_g$.

Example

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{$price 2^i for $i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $\sum_{g \in \mathcal{G}} h_g = \sum_i^{\log h} 2^i$.

Example

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{$price 2^i for $i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $\sum_{g \in \mathcal{G}} h_g = \sum_i^{\log h} 2^i \leq 2h$.

Example

Example: Selling tee shirts. (discretized prices)

- Bidders with valuations in $[1, h]$ for a tee shirt.
- Reasonable offers: $\mathcal{G} = \{\text{price } 2^i \text{ for } i \in \{1, \ldots, \log h\}\}$.
- Convergence Time: $\sum_{g \in \mathcal{G}} h_g = \sum_i^{\log h} 2^i \leq 2h$.

Note: this is optimal up to constant factors.

Conclusions

1. Used machine learning techniques for auction design/analysis.
2. Prior-free discriminatory optimal mechanism design.
   (a) distinguishing between products (and selecting products to sell).
   (b) price discriminate based on observable customer features.
3. Similar bounds for offline and online auctions.
4. Retrospective analysis for offline auctions.

Conclusions

1. Used machine learning techniques for auction design/analysis.
2. Prior-free discriminatory optimal mechanism design.
   (a) distinguishing between products (and selecting products to sell).
   (b) price discriminate based on observable customer features.
3. Similar bounds for offline and online auctions.
4. Retrospective analysis for offline auctions.
5. **Open:** ϵ-cover arguments for online auctions?
6. **Open:** limited supply?
7. **Open:** general cost function on outcomes?