The Simple Mathematics of Optimal Auctions

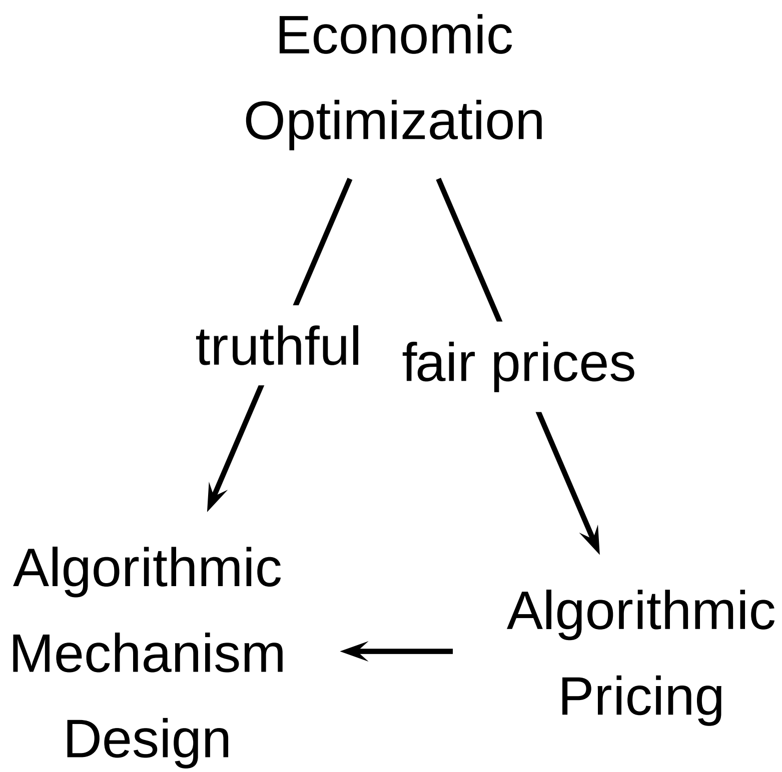
Jason D. Hartline

(joint with Maria-Florina Balcan,

Nikhil Devanur, and Kunal Talwar)

March 28, 2007

Economic Optimization

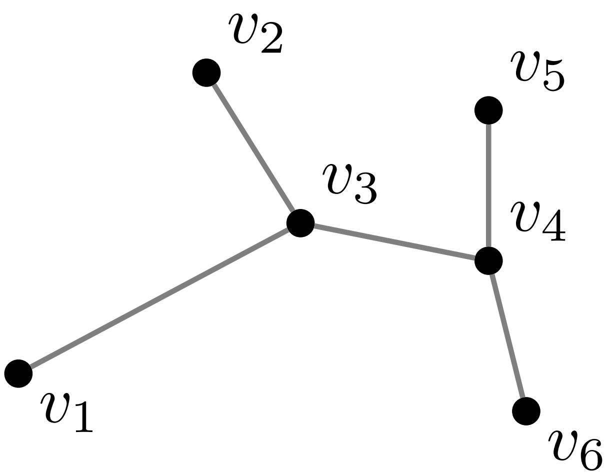


Overview

$\Longrightarrow$ 1. Review unlimited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.
2. Generalize to limited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.
3. Generality & conclusions.

Example: Path Pricing

Example: *Edge pricing selling paths.*

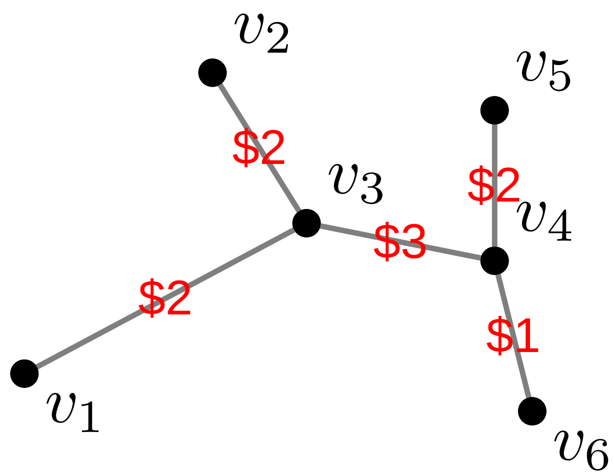


Consumer 1 wants path from v_1 to v_2 for \$5.
Consumer 2 wants path from v_2 to v_3 for \$3.

$\vdots$

Example: Path Pricing

Example: *Edge pricing selling paths.*

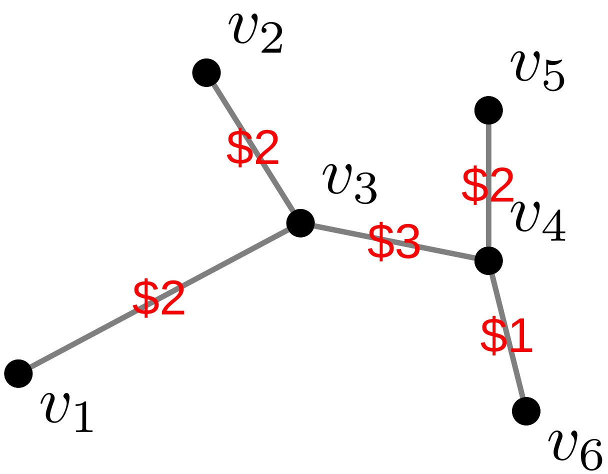


Consumer 1 wants path from v_1 to v_2 for $5.
Consumer 2 wants path from v_2 to v_3 for $3.

$\vdots$

Example: Path Pricing

Example: *Edge pricing selling paths.*

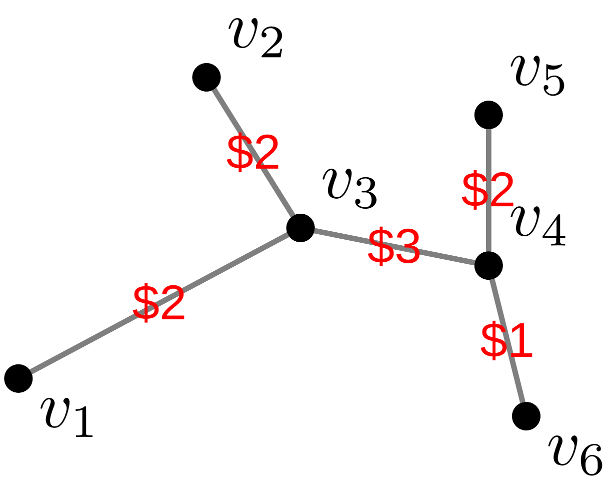


Consumer 1 wants path from v_1 to v_2 for \$5. (pays \$4)
Consumer 2 wants path from v_2 to v_3 for \$3.

$\vdots$

Example: Path Pricing

Example: *Edge pricing selling paths.*

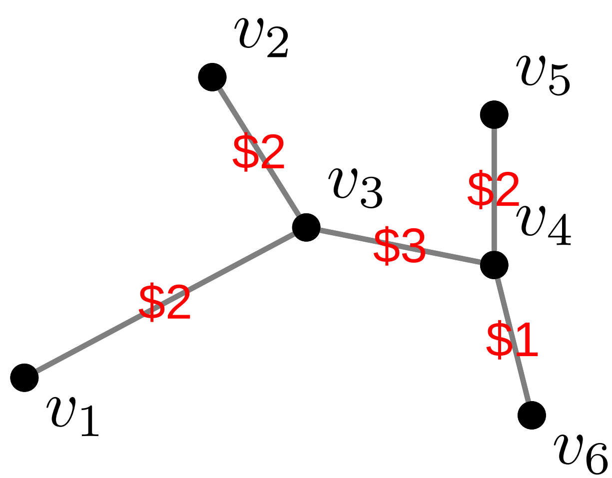


Consumer 1 wants path from v_1 to v_2 for \$5. (pays \$4)
Consumer 2 wants path from v_2 to v_3 for \$3. (not served)

$\vdots$

Example: Path Pricing

Example: *Edge pricing selling paths.*



Consumer 1 wants path from v_1 to v_2 for \$5. (pays \$4)
Consumer 2 wants path from v_2 to v_3 for \$3. (not served)

$\vdots$

Goal: price edges to maximize objective.

Unlimited Supply Algorithmic Pricing

The *Unlimited Supply Algorithmic Pricing* problem:

Given:

- unlimited supply of stuff.
- Set S of n consumers and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Unlimited Supply Algorithmic Pricing

The *Unlimited Supply Algorithmic Pricing* problem:

Given:

- unlimited supply of stuff.
- Set S of n consumers and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

- $p(i, g) =$ payoff from consumer i when offered $g \in \mathcal{G}$.

Unlimited Supply Algorithmic Pricing

The *Unlimited Supply Algorithmic Pricing* problem:

Given:

- unlimited supply of stuff.
- Set S of n consumers and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

- $p(i,g) =$ payoff from consumer i when offered $g \in \mathcal{G}$.
- $p(S,g) = \sum_{i \in S} p(i,g)$.

Unlimited Supply Algorithmic Pricing

The *Unlimited Supply Algorithmic Pricing* problem:

Given:

- unlimited supply of stuff.
- Set S of n consumers and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

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

Unlimited Supply Algorithmic Pricing

The *Unlimited Supply Algorithmic Pricing* problem:

Given:

- unlimited supply of stuff.
- Set S of n consumers and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

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

Example: Digital Good

Example: *digital good*

- Single item for sale (unlimited supply).
- Consumers have valuations for single copy of item, $(v_1, \ldots, v_n)$.
- Consumers are indistinguishable.

Example: Digital Good

Example: *digital good*

- Single item for sale (unlimited supply).
- Consumers have valuations for single copy of item, $(v_1, \ldots, v_n)$.
- Consumers are indistinguishable.
- $\mathcal{G} =$ set of all prices, i.e., $g_q =$ "take-it-or-leave-it at price q".

Example: *digital good*

- Single item for sale (unlimited supply).
- Consumers have valuations for single copy of item, $(v_1, \ldots, v_n)$.
- Consumers are indistinguishable.
- $\mathcal{G}$ = set of all prices, i.e., g_q = "take-it-or-leave-it at price q".
- $p(i, g_q) = \begin{cases} q & \text{if } q \leq v_i \\ 0 & \text{o.w.} \end{cases}$.

Example: Digital Good

Example: *digital good*

- Single item for sale (unlimited supply).
- Consumers have valuations for single copy of item, $(v_1, \ldots, v_n)$.
- Consumers are indistinguishable.
- $\mathcal{G} =$ set of all prices, i.e., $g_q =$ "take-it-or-leave-it at price q".
- $p(i, g_q) = \begin{cases} q & \text{if } q \leq v_i \\ 0 & \text{o.w.} \end{cases}$.

How can we compute $\mathrm{opt}_{\mathcal{G}}$?

Example: Digital Good

Example: *digital good*

- Single item for sale (unlimited supply).
- Consumers have valuations for single copy of item, $(v_1, \ldots, v_n)$.
- Consumers are indistinguishable.
- $\mathcal{G}$ = set of all prices, i.e., g_q = "take-it-or-leave-it at price q".
- $p(i, g_q) = \begin{cases} q & \text{if } q \leq v_i \\ 0 & \text{o.w.} \end{cases}$.

How can we compute $\mathrm{opt}_{\mathcal{G}}$?

1. Sort valuations: $v_1 \geq \ldots \geq v_n$
2. Output v_i to maximize $i \times v_i$.

Literature

Algorithmic Pricing in the Literature

- unlimited supply (mostly).
- many interesting special cases.
- includes work of: Gagan Aggarwal, Maria-Florina Balcan, Avrim Blum, Patrick Briest, Shuchi Chawla, Eric Demaine, Tomás Feder, Uri Feige, Venkat Gurusuami, MohammadTaghi Hajiaghayi, Anna Karlin, David Kempe, Vladlin Koltun, Robert Kleinberg, Piotr Krysta, Clare Mathieu, Frank McSherry, Rajeev Motwani, and An Zhu.

Literature

Algorithmic Pricing in the Literature

- unlimited supply (mostly).
- many interesting special cases.
- includes work of: Gagan Aggarwal, Maria-Florina Balcan, Avrim Blum, Patrick Briest, Shuchi Chawla, Eric Demaine, Tomás Feder, Uri Feige, Venkat Gurusuami, MohammadTaghi Hajiaghayi, Anna Karlin, David Kempe, Vladlin Koltun, Robert Kleinberg, Piotr Krysta, Clare Mathieu, Frank McSherry, Rajeev Motwani, and An Zhu.
- hard (even to approximate).

Overview

1. Review unlimited supply setting:
   (a) Algorithmic pricing.
   $\Longrightarrow$ (b) Mechanism design via pricing.
2. Generalize to limited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.
3. Generality & conclusions.

Auction Problem

The *Unlimited Supply Auction Problem*:

Given:

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

Design: Single round, sealed bid, *truthful* auction with profit near that of $\mathrm{OPT}_{\mathcal{G}}$.

Recall Notation:

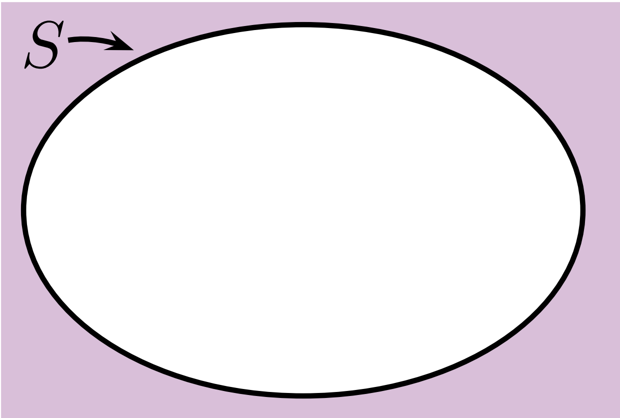
- $g(i) =$ payoff from bidder i when offered g.
- $g(S) = \sum_{i \in S} g(i)$.
- $\mathrm{opt}_{\mathcal{G}}(S) = \mathrm{argmax}_{g \in \mathcal{G}}\, g(S)$.
- $\mathrm{OPT} = \mathrm{OPT}_{\mathcal{G}}(S) = \max_{g \in \mathcal{G}}\, g(S)$.

Random Sampling Auction

Generalization of auction from [Goldberg, Hartline, Wright '01]:

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.

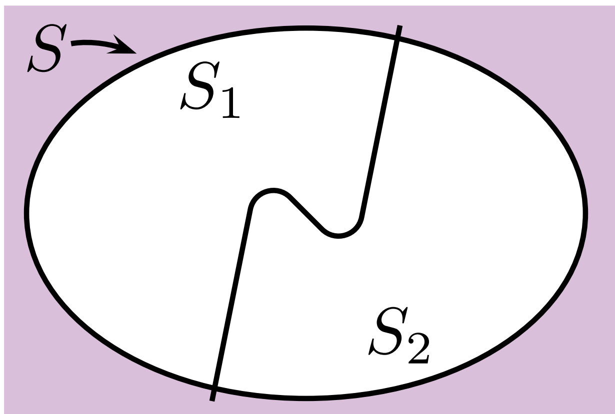


Random Sampling Auction

Generalization of auction from [Goldberg, Hartline, Wright '01]:

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.

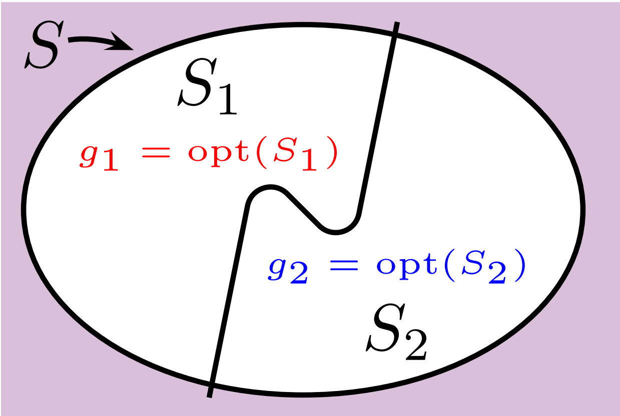


Random Sampling Auction

Generalization of auction from [Goldberg, Hartline, Wright '01]:

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.

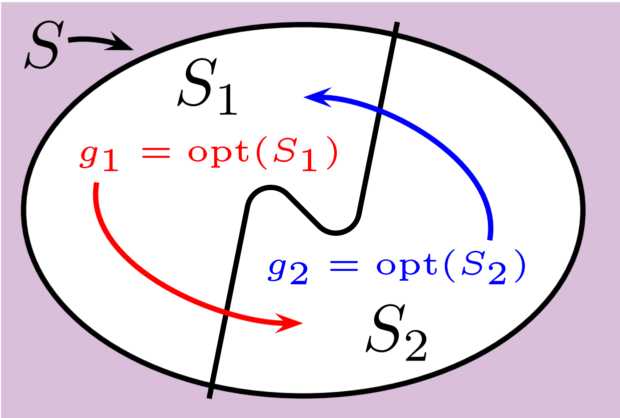


Random Sampling Auction

Generalization of auction from [Goldberg, Hartline, Wright '01]:

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.

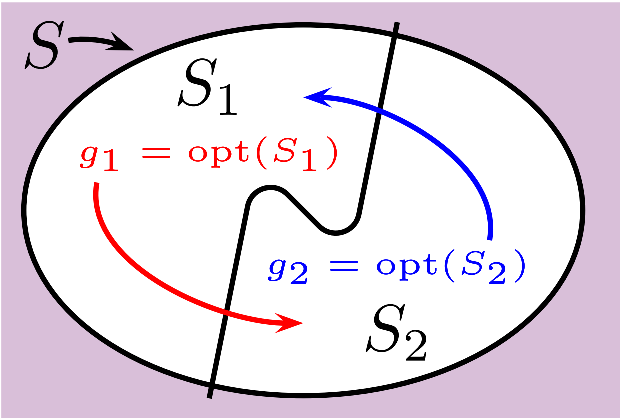


Random Sampling Auction

Generalization of auction from [Goldberg, Hartline, Wright '01]:

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.



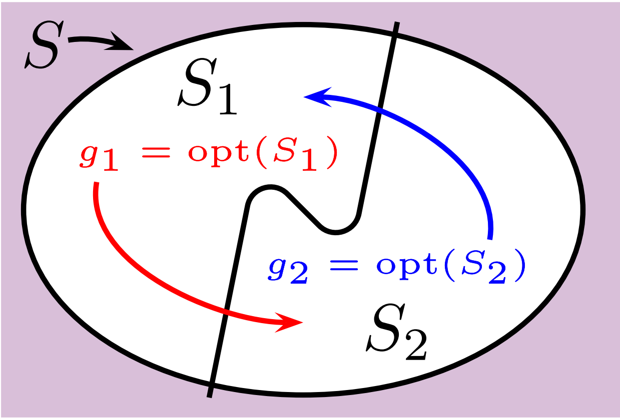
Fact: $\mathrm{RSOO}_{\mathcal{G}}$ is truthful.

Random Sampling Auction

Generalization of auction from [Goldberg, Hartline, Wright '01]:

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.



Fact: RSOO$_{\mathcal{G}}$ is truthful.

Question: when does RSOO$_{\mathcal{G}}$ perform well?

Performance Analysis

(The following analysis is from [Balcan, Blum, Hartline, Mansour ’05])

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon \, \mathrm{OPT}.$$

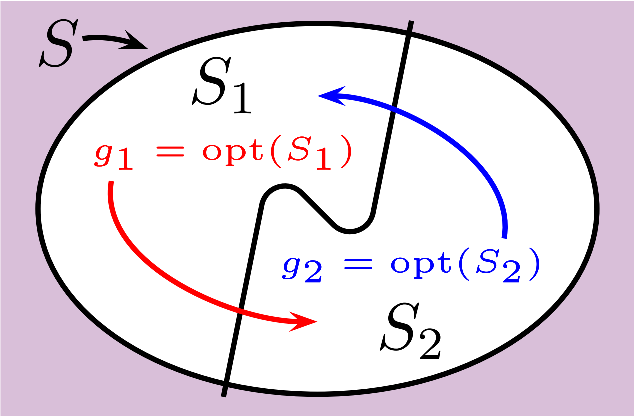
Performance Analysis

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon\,\mathrm{OPT}.$$

Intuition:

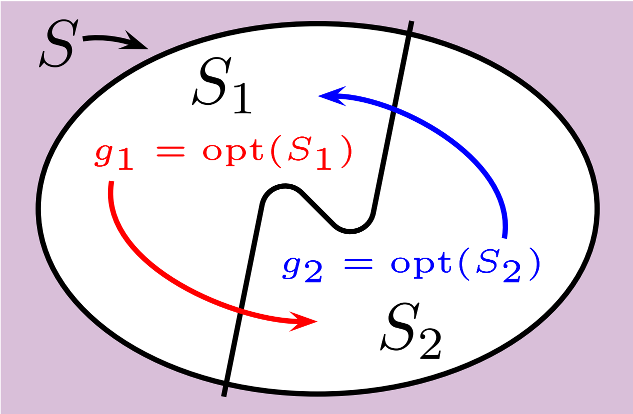


(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon \, \mathrm{OPT}.$$

Intuition:



- Suppose all $g \in \mathcal{G}$ are *good*, then

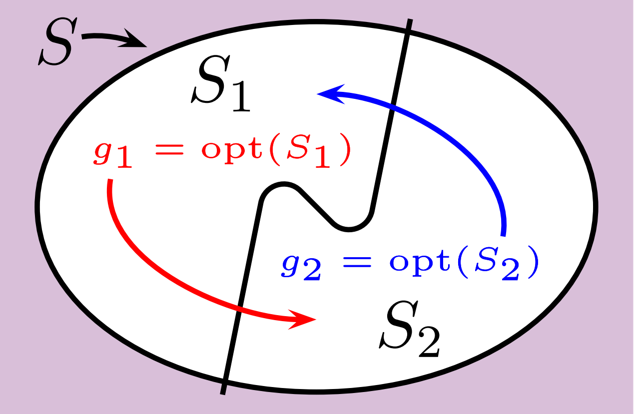
Performance Analysis

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon\,\mathrm{OPT}.$$

Intuition:



- Suppose all $g \in \mathcal{G}$ are *good*, then
- $p(S_1, g_2) \geq p(S_2, g_2) - \epsilon\,\mathrm{OPT}_{\mathcal{G}}.$
  $p(S_2, g_1) \geq p(S_1, g_1) - \epsilon\,\mathrm{OPT}_{\mathcal{G}}.$

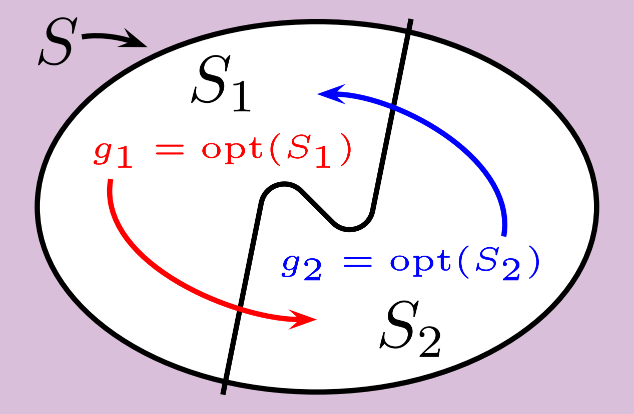
Performance Analysis

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon \, \mathrm{OPT}.$$

Intuition:



- Suppose all $g \in \mathcal{G}$ are *good*, then
- $p(S_1, g_2) \geq p(S_2, g_2) - \epsilon \, \mathrm{OPT}_{\mathcal{G}}$.
  $p(S_2, g_1) \geq p(S_1, g_1) - \epsilon \, \mathrm{OPT}_{\mathcal{G}}$.
- $p(S_1, g_1) + p(S_2, g_2) \geq \mathrm{OPT}_{\mathcal{G}}$.

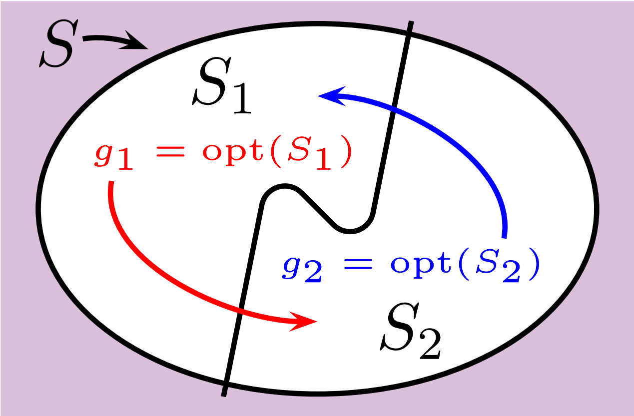
Performance Analysis

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon\,\mathrm{OPT}.$$

Intuition:



- Suppose all $g \in \mathcal{G}$ are *good*, then
- $p(S_1, g_2) \geq p(S_2, g_2) - \epsilon\,\mathrm{OPT}_{\mathcal{G}}$.
  $p(S_2, g_1) \geq p(S_1, g_1) - \epsilon\,\mathrm{OPT}_{\mathcal{G}}$.
- $p(S_1, g_1) + p(S_2, g_2) \geq \mathrm{OPT}_{\mathcal{G}} = p(S_1, g^*) + p(S_2, g^*)$.

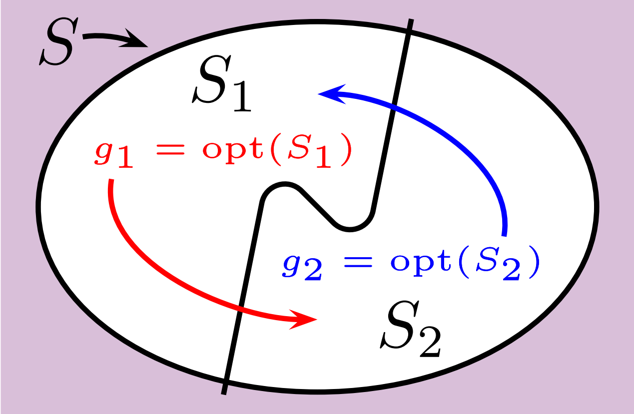
Performance Analysis

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon \, \mathrm{OPT}.$$

Intuition:



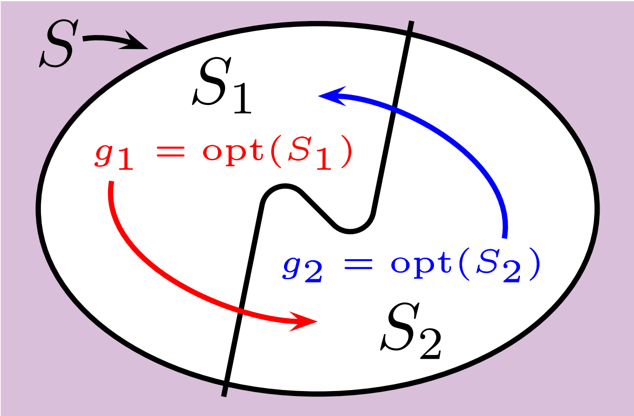
- Suppose all $g \in \mathcal{G}$ are *good*, then
- $p(S_1, g_2) \geq p(S_2, g_2) - \epsilon \, \mathrm{OPT}_{\mathcal{G}}$.
  $p(S_2, g_1) \geq p(S_1, g_1) - \epsilon \, \mathrm{OPT}_{\mathcal{G}}$.
- $p(S_1, g_1) + p(S_2, g_2) \geq \mathrm{OPT}_{\mathcal{G}} = p(S_1, g^*) + p(S_2, g^*)$.
- Profit $= p(S_1, g_2) + p(S_2, g_1)$.

(The following analysis is from [Balcan, Blum, Hartline, Mansour '05])

Definition: g is *good* for partitions S_1 and S_2 if

$$|p(S_1, g) - p(S_2, g))| \leq \epsilon \, \mathrm{OPT}.$$

Intuition:



- Suppose all $g \in \mathcal{G}$ are *good*, then
- $p(S_1, g_2) \geq p(S_2, g_2) - \epsilon \, \mathrm{OPT}_{\mathcal{G}}$.
  $p(S_2, g_1) \geq p(S_1, g_1) - \epsilon \, \mathrm{OPT}_{\mathcal{G}}$.
- $p(S_1, g_1) + p(S_2, g_2) \geq \mathrm{OPT}_{\mathcal{G}} = p(S_1, g^*) + p(S_2, g^*)$.
- Profit $= p(S_1, g_2) + p(S_2, g_1) \geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon \, \mathrm{OPT}$.

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon\,\mathrm{OPT}$.

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon\,\mathrm{OPT}$.

Lemma: For g with $g(i) \leq h$ and random partitions S_1 and S_2:

$$\mathbf{Pr}[g \text{ not good}] \leq 2e^{-\epsilon^2 \mathrm{OPT}/2h}.$$

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon\,\mathrm{OPT}$.

Lemma: For g with $g(i) \leq h$ and random partitions S_1 and S_2:

$$\textbf{Pr}[g \text{ not good}] \leq 2e^{-\epsilon^2\,\mathrm{OPT}\,/2h}.$$

Consider: (for $\delta \ll 1$)

- Suppose: $|\mathcal{G}|\, e^{-\epsilon^2\,\mathrm{OPT}\,/2h} \leq \delta$.

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \text{OPT}_{\mathcal{G}} - 2\epsilon \,\text{OPT}$.

Lemma: For g with $g(i) \leq h$ and random partitions S_1 and S_2:

$$\textbf{Pr}[g \text{ not good}] \leq 2e^{-\epsilon^2 \,\text{OPT}\,/2h}.$$

Consider: (for $\delta \ll 1$)

- Suppose: $|\mathcal{G}|\, e^{-\epsilon^2 \,\text{OPT}\,/2h} \leq \delta$.
- Then: union bound $\Rightarrow$ **Pr**$[$any $g \in \mathcal{G}$ is not good$] \leq \delta$.

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon\,\mathrm{OPT}$.

Lemma: For g with $g(i) \leq h$ and random partitions S_1 and S_2:

$$\textbf{Pr}[g \text{ not good}] \leq 2e^{-\epsilon^2\,\mathrm{OPT}/2h}.$$

Consider: (for $\delta \ll 1$)

- Suppose: $|\mathcal{G}|\, e^{-\epsilon^2\,\mathrm{OPT}/2h} \leq \delta$. (i.e., $\mathrm{OPT}_{\mathcal{G}} \geq \frac{2h}{\epsilon^2} \ln \frac{|\mathcal{G}|}{\delta}$)
- Then: union bound $\Rightarrow$ **Pr**$[$any $g \in \mathcal{G}$ is not good$] \leq \delta$.

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon\,\mathrm{OPT}$.

Lemma: For g with $g(i) \leq h$ and random partitions S_1 and S_2:

$$\mathbf{Pr}[g \text{ not good}] \leq 2e^{-\epsilon^2\,\mathrm{OPT}/2h}.$$

Consider: (for $\delta \ll 1$)

- Suppose: $|\mathcal{G}|\, e^{-\epsilon^2\,\mathrm{OPT}/2h} \leq \delta$. (i.e., $\mathrm{OPT}_{\mathcal{G}} \geq \frac{2h}{\epsilon^2} \ln \frac{|\mathcal{G}|}{\delta}$)
- Then: union bound $\Rightarrow$ **Pr**[any $g \in \mathcal{G}$ is not good] $\leq \delta$.

Theorem: With probability $1 - \delta$,

$$\text{Profit} \geq (1 - 2\epsilon)\,\mathrm{OPT}_{\mathcal{G}} \quad \text{when} \quad \mathrm{OPT}_{\mathcal{G}} \geq \frac{2h}{\epsilon^2} \log \frac{|\mathcal{G}|}{\delta}.$$

Performance Analysis (cont)

Lemma: All $g \in \mathcal{G}$ are good $\Rightarrow$ Profit $\geq \mathrm{OPT}_{\mathcal{G}} - 2\epsilon\,\mathrm{OPT}$.

Lemma: For g with $g(i) \leq h$ and random partitions S_1 and S_2:

$$\mathbf{Pr}[g \text{ not good}] \leq 2e^{-\epsilon^2\,\mathrm{OPT}/2h}.$$

Consider: (for $\delta \ll 1$)

- Suppose: $|\mathcal{G}|\, e^{-\epsilon^2\,\mathrm{OPT}/2h} \leq \delta$. (i.e., $\mathrm{OPT}_{\mathcal{G}} \geq \frac{2h}{\epsilon^2} \ln \frac{|\mathcal{G}|}{\delta}$)
- Then: union bound $\Rightarrow$ **Pr**[any $g \in \mathcal{G}$ is not good] $\leq \delta$.

Theorem: With probability $1 - \delta$,

$$\text{Profit} \geq (1 - 2\epsilon)\,\mathrm{OPT}_{\mathcal{G}} \quad \text{when} \quad \mathrm{OPT}_{\mathcal{G}} \geq \frac{2h}{\epsilon^2} \log \frac{|\mathcal{G}|}{\delta}.$$

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

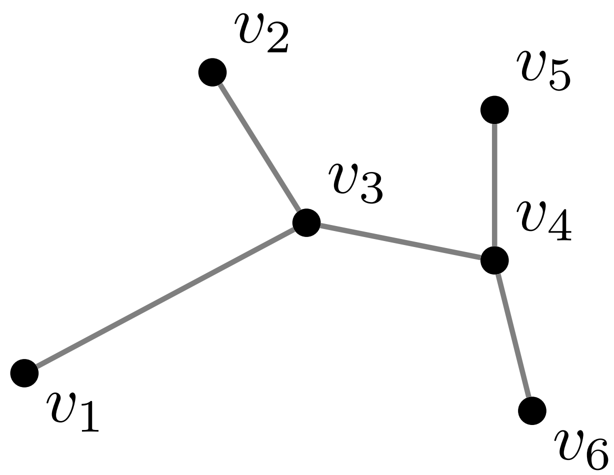
Example: Digital Good Auctions

Example: Digital good with discretized prices.

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

Example: Path Auctions

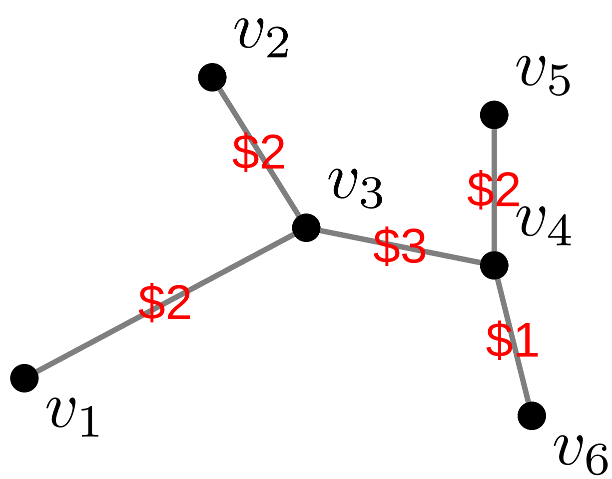
E.g., selling bandwidth on paths in a graph.



Consumer 1 wants path from v_1 to v_2 for \$5.
Consumer 2 wants path from v_2 to v_3 for \$3.
...
Consumer n wants path from v_1 to v_5 for \$6.

Example: Path Auctions

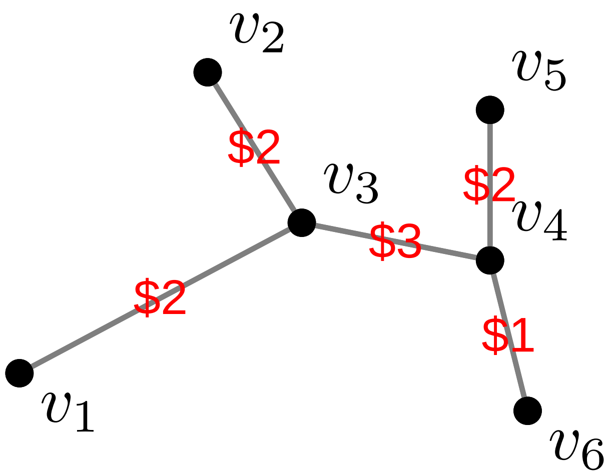
E.g., selling bandwidth on paths in a graph.



Consumer 1 wants path from v_1 to v_2 for \$5.
Consumer 2 wants path from v_2 to v_3 for \$3.
...
Consumer n wants path from v_1 to v_5 for \$6.

Example: Path Auctions

E.g., selling bandwidth on paths in a graph.

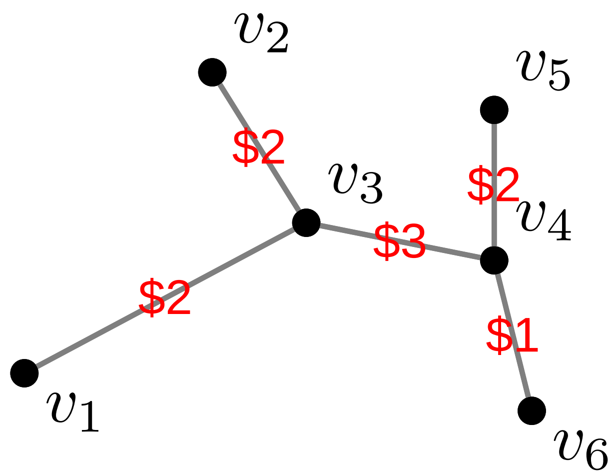


Consumer 1 wants path from v_1 to v_2 for \$5.
Consumer 2 wants path from v_2 to v_3 for \$3.
...
Consumer n wants path from v_1 to v_5 for \$6.

Let $\mathcal{G}$ be set of power-of-two pricings of links in the network.

Example: Path Auctions

E.g., selling bandwidth on paths in a graph.



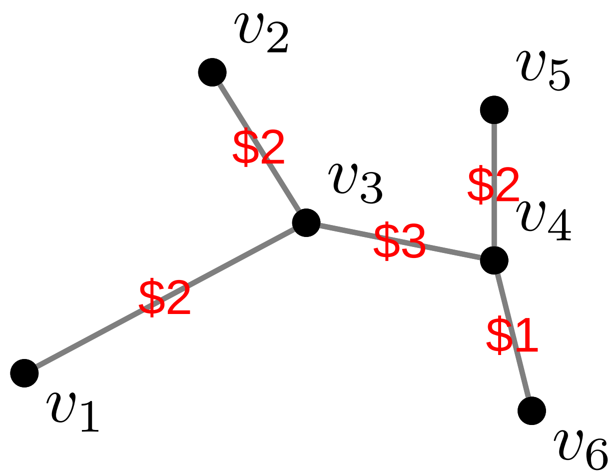
Consumer 1 wants path from v_1 to v_2 for $5.
Consumer 2 wants path from v_2 to v_3 for $3.
. . .
Consumer n wants path from v_1 to v_5 for $6.

Let $\mathcal{G}$ be set of power-of-two pricings of links in the network.

Fact: For network with m links, $|\mathcal{G}| \approx \log^m h$

Example: Path Auctions

E.g., selling bandwidth on paths in a graph.



Consumer 1 wants path from v_1 to v_2 for \$5.
Consumer 2 wants path from v_2 to v_3 for \$3.
. . .
Consumer n wants path from v_1 to v_5 for \$6.

Let $\mathcal{G}$ be set of power-of-two pricings of links in the network.

Fact: For network with m links, $|\mathcal{G}| \approx \log^m h$

Result: Convergence rate of RSOO$_{\mathcal{G}}$ is $O(hm \log\log h)$.

Overview

1. Review unlimited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.

$\Longrightarrow$ 2. Generalize to limited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.

3. Generality & conclusions.

Limited Supply Algorithmic Pricing

The *Limited Supply Algorithmic Pricing* problem:

Given:

- limited supply of stuff, $C_1, \ldots, C_m$
- Set S of n bidders and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

The *Limited Supply Algorithmic Pricing* problem:

Given:

- limited supply of stuff, $C_1, \ldots, C_m$
- Set S of n bidders and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

- $p(i, g) =$ payoff from consumer i when offered $g \in \mathcal{G}$.
- $p(S, g) = \sum_{i \in S} p(i, g)$.

Limited Supply Algorithmic Pricing

The *Limited Supply Algorithmic Pricing* problem:

Given:

- limited supply of stuff, $C_1, \ldots, C_m$
- Set S of n bidders and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

- $p(i, g) =$ payoff from consumer i when offered $g \in \mathcal{G}$.
- $p(S, g) = \sum_{i \in S} p(i, g)$.
- $x_j(i, g) =$ consumer i's demand for item j when offered $g \in \mathcal{G}$.
- $x_j(S, g) = \sum_{i \in S} x_j(i, g)$.

Limited Supply Algorithmic Pricing

The *Limited Supply Algorithmic Pricing* problem:

Given:

- limited supply of stuff, $C_1, \ldots, C_m$
- Set S of n bidders and their preferences for stuff.
- class $\mathcal{G}$ of reasonable offers.

Design: Algorithm to compute optimal offer from $\mathcal{G}$.

Notation:

- $p(i, g) =$ payoff from consumer i when offered $g \in \mathcal{G}$.
- $p(S, g) = \sum_{i \in S} p(i, g)$.
- $x_j(i, g) =$ consumer i's demand for item j when offered $g \in \mathcal{G}$.
- $x_j(S, g) = \sum_{i \in S} x_j(i, g)$.

What if $x_j(S, g) > C_j$?

Dealing with Excess Demand

Two approaches:

- restrict algorithm. [Gurusuami et al. '05]
  - i.e., only consider $g \in \mathcal{G}$ with $x_j(S, g) \leq C_j$ for all j)

Dealing with Excess Demand

Two approaches:

- restrict algorithm. [Gurusuami et al. '05]
  - i.e., only consider $g \in \mathcal{G}$ with $x_j(S, g) \leq C_j$ for all j)
  - problem: random sampling auction may still exceed supply.

Dealing with Excess Demand

Two approaches:

- restrict algorithm. [Gurusuami et al. '05]
  - i.e., only consider $g \in \mathcal{G}$ with $x_j(S, g) \leq C_j$ for all j)
  - problem: random sampling auction may still exceed supply.
- prioritize consumers randomly. [Borgs et al. '05]
  - randomly order bidders
  - make offer "first come first served, while supplies last"

Dealing with Excess Demand

Two approaches:

- restrict algorithm. [Gurusuami et al. '05]
  - i.e., only consider $g \in \mathcal{G}$ with $x_j(S, g) \leq C_j$ for all j)
  - problem: random sampling auction may still exceed supply.
- prioritize consumers randomly. [Borgs et al. '05]
  - randomly order bidders
  - make offer "first come first served, while supplies last"

  What is the payoff of offer g?

Single Commodity and Uniform Knapsack

A knapsack problem:

- consumer payoffs: $p(1, g), \ldots, p(n, g)$.
- consumer demands: $x(1, g), \ldots, x(n, g)$.
- capacity: C

Question: what is expected payoff of “random first come first served”?

Single Commodity and Uniform Knapsack

A knapsack problem:

- consumer payoffs: $p(1,g),\ldots,p(n,g)$.
- consumer demands: $x(1,g),\ldots,x(n,g)$.
- capacity: C

Question: what is expected payoff of “random first come first served”?

Theorem: When $x(i,S) > C$ then

$$\mathbf{E}[\text{Payoff}(S,g,C)] = \frac{(C \pm \Theta(x_{\max}))p(S,g)}{x(i,S)}$$

where $x_{\max} = \max_i x(i,g)$.

Single Commodity and Uniform Knapsack

A knapsack problem:

- consumer payoffs: $p(1,g),\ldots,p(n,g)$.
- consumer demands: $x(1,g),\ldots,x(n,g)$.
- capacity: C

Question: what is expected payoff of "random first come first served"?

Theorem: When $x(i,S) > C$ then

$$\mathbf{E}[\text{Payoff}(S,g,C)] = \frac{(C \pm \Theta(x_{\max}))p(S,g)}{x(i,S)}$$

where $x_{\max} = \max_i x(i,g)$.

Proof: via reduction to uniform payoff case (i.e., $p(i,g) = 1$)

Definition: *Estimated payoff* of g on S: $P(S,g,C) = \frac{C \cdot p(S,g)}{\max\{C, x(S,g)\}}$

Limited Supply Algorithmic Pricing

Definition: *Estimated payoff* of g on S: $P(S,g,C) = \frac{C \cdot p(S,g)}{\max\{C, x(S,g)\}}$

- $\mathrm{opt}_{\mathcal{G}}(S,C) = \mathrm{argmax}_{g \in \mathcal{G}}\, P(S,g,C)$.
- $\mathrm{OPT}_{\mathcal{G}}(S,C) = \max_{g \in \mathcal{G}}\, P(S,g,C)$.

Algorithmic Pricing Goal: compute $\mathrm{opt}_{\mathcal{G}}(S,C)$.

Overview

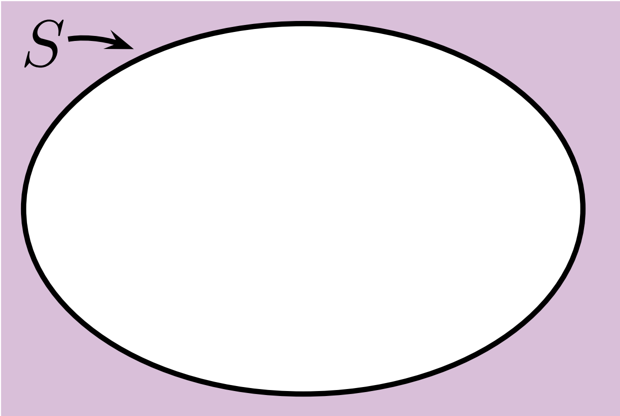
1. Review unlimited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.
2. Generalize to limited supply setting:
   (a) Algorithmic pricing.
   $\Longrightarrow$ (b) Mechanism design via pricing.
3. Generality & conclusions.

Limited Supply Random Sampling Auction

Generalization of auction from [Borgs et al. '05]:

Random Sampling Limited Supply Auction, RSLS$_{\mathcal{G}}$

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) on half supply.
3. Offer g_1 to S_2 and g_2 to S_1.

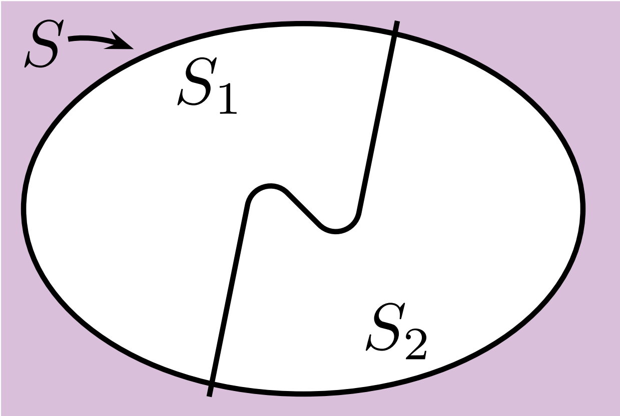


Limited Supply Random Sampling Auction

Generalization of auction from [Borgs et al. '05]:

Random Sampling Limited Supply Auction, RSLS$_{\mathcal{G}}$

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) on half supply.
3. Offer g_1 to S_2 and g_2 to S_1.

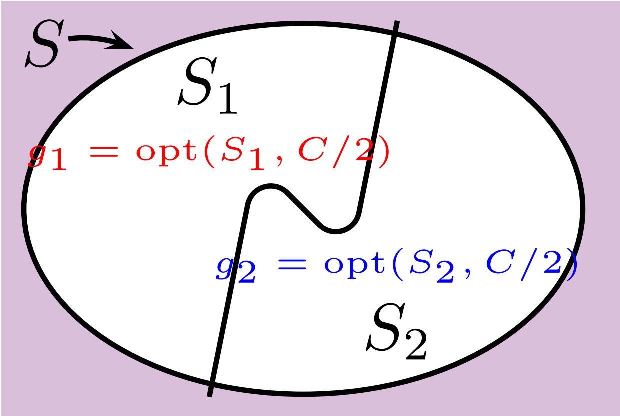


Limited Supply Random Sampling Auction

Generalization of auction from [Borgs et al. '05]:

Random Sampling Limited Supply Auction, RSLS$_{\mathcal{G}}$

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) on half supply.
3. Offer g_1 to S_2 and g_2 to S_1.

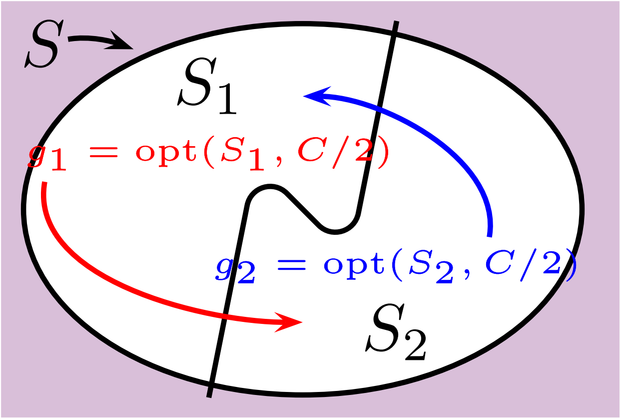


Limited Supply Random Sampling Auction

Generalization of auction from [Borgs et al. '05]:

Random Sampling Limited Supply Auction, RSLS$_{\mathcal{G}}$

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) on half supply.
3. Offer g_1 to S_2 and g_2 to S_1.

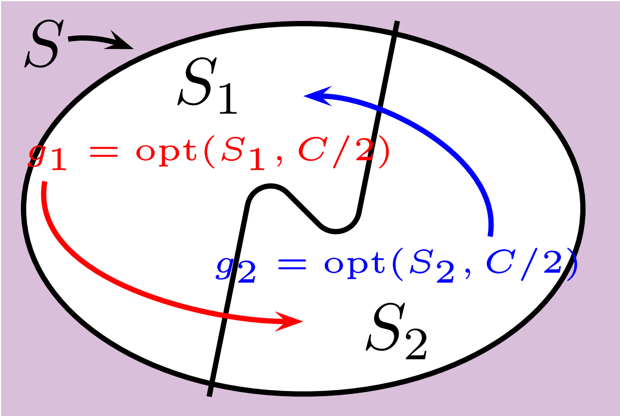


Limited Supply Random Sampling Auction

Generalization of auction from [Borgs et al. '05]:

Random Sampling Limited Supply Auction, RSLS$_{\mathcal{G}}$

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) on half supply.
3. Offer g_1 to S_2 and g_2 to S_1.



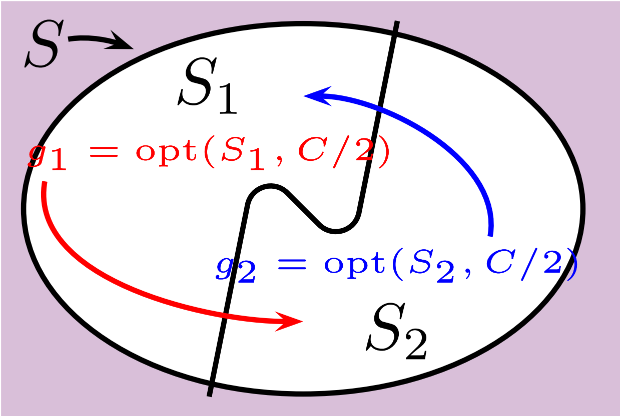
Fact: RSLS$_{\mathcal{G}}$ is truthful.

Limited Supply Random Sampling Auction

Generalization of auction from [Borgs et al. '05]:

Random Sampling Limited Supply Auction, RSLS$_\mathcal{G}$

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) on half supply.
3. Offer g_1 to S_2 and g_2 to S_1.



Fact: RSLS$_\mathcal{G}$ is truthful.

Question: when does RSLS$_\mathcal{G}$ perform well?

RSLS$_{\mathcal{G}}$ Performance

Theorem: With probability $1 - \delta$,

$$\text{Profit} \geq (1 - \epsilon)\,\mathrm{OPT}_{\mathcal{G}}$$

when $\frac{\mathrm{OPT}_{\mathcal{G}}}{p_{\max}}$ and $\frac{C}{x_{\max}}$ are $O(\frac{1}{\epsilon^2}\log\frac{4|\mathcal{G}|}{\delta})$.

Proof Sketch:

1. With probability $1 - \delta$ all g are *ϵ-good*.
   (with respect to $p(S, g)$ and $x(S, g)$).
2. Thus, g_1 and g_2 are ϵ-good.
3. $P(S_1, g_2, C/2) \geq (1 - \epsilon')P(S_2, g_2, C/2)$.
4. $P(S_1, g^*, C/2) + P(S_1, g^*, C/2) \geq (1 - \epsilon'')P(S, g^*, C)$.
5. Profit $\geq (1 - \epsilon''')\,\mathrm{OPT}_{\mathcal{G}}(S, C)$.

Example Analysis:

Claim: $P(S_1, g_2, C/2) \geq (1 - \epsilon')P(S_2, g_2, C/2)$.

Sketch:

$$
\begin{aligned}
P(S_1, g_2, C/2) &= \tfrac{C}{2}\frac{p(S_1, g_2)}{\max\{C/2, x(S_1, g_2)\}}) \\
&\geq \tfrac{C}{2}\frac{(1-\epsilon)p(S_2, g_2)}{(1+\epsilon)\max\{C/2, x(S_2, g_2)\}} \\
&= (1 - 2\epsilon)P(S_2, g_2, C/2).
\end{aligned}
$$

Example Analysis:

Claim: $P(S_1, g_2, C/2) \geq (1-\epsilon')P(S_2, g_2, C/2)$.

Sketch:

$$\begin{aligned} P(S_1, g_2, C/2) &= \tfrac{C}{2}\frac{p(S_1, g_2)}{\max\{C/2, x(S_1, g_2)\}}) \\ &\geq \tfrac{C}{2}\frac{(1-\epsilon)p(S_2, g_2)}{(1+\epsilon)\max\{C/2, x(S_2, g_2)\}} \\ &= (1-2\epsilon)P(S_2, g_2, C/2). \end{aligned}$$

Key Fact for Theorem: $p(S, g)$ and $x(S, g)$ are sums of i.i.d. variables.

Overview

$\Longrightarrow$ 3. Generality & conclusions.

1. Review unlimited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.
2. Generalize to limited supply setting:
   (a) Algorithmic pricing.
   (b) Mechanism design via pricing.
3. Generality & conclusions.

Generality

This approach is very general:

Generality

This approach is very general:

- General linear objectives: $p(S, g) = \sum_{i \in S} p(i, g)$

Generality

This approach is very general:

- General linear objectives: $p(S,g) = \sum_{i \in S} p(i,g)$
  - maximize profit (i.e., $p(i,g) =$ payment)
  - maximize welfare (i.e., $p(i,g) =$ value)

Generality

This approach is very general:

- General linear objectives: $p(S, g) = \sum_{i \in S} p(i, g)$
  - maximize profit (i.e., $p(i, g) =$ payment)
  - maximize welfare (i.e., $p(i, g) =$ value)
- General agent preferences:
  (given g, agent chooses favorite outcome, price)

Generality

This approach is very general:

- General linear objectives: $p(S, g) = \sum_{i \in S} p(i, g)$
  - maximize profit (i.e., $p(i, g) =$ payment)
  - maximize welfare (i.e., $p(i, g) =$ value)
- General agent preferences:
  (given g, agent chooses favorite outcome, price)
  - quasi-linear:
    utility(outcome,price) $=$ value(outcome) $-$ price.

Generality

This approach is very general:

- General linear objectives: $p(S, g) = \sum_{i \in S} p(i, g)$
  - maximize profit (i.e., $p(i, g) =$ payment)
  - maximize welfare (i.e., $p(i, g) =$ value)
- General agent preferences:
  (given g, agent chooses favorite outcome, price)
  - quasi-linear:
    utility(outcome,price) $=$ value(outcome) $-$ price.
  - budgets:
    utility(outcome,price)
    $= \begin{cases} \infty & \text{if price} > \text{budget} \\ \text{value(outcome) - price} & \text{otherwise} \end{cases}$

Generality

This approach is very general:

- General linear objectives: $p(S, g) = \sum_{i \in S} p(i, g)$
  - maximize profit (i.e., $p(i, g) =$ payment)
  - maximize welfare (i.e., $p(i, g) =$ value)
- General agent preferences:
  (given g, agent chooses favorite outcome, price)
  - quasi-linear:
    utility(outcome,price) $=$ value(outcome) $-$ price.
  - budgets:
    utility(outcome,price)
    $= \begin{cases} \infty & \text{if price} > \text{budget} \\ \text{value(outcome) - price} & \text{otherwise} \end{cases}$
  - etc.

Generality

This approach is very general:

- General linear objectives: $p(S,g) = \sum_{i \in S} p(i,g)$
  - maximize profit (i.e., $p(i,g) =$ payment)
  - maximize welfare (i.e., $p(i,g) =$ value)
- General agent preferences:
  (given g, agent chooses favorite outcome, price)
  - quasi-linear:
    utility(outcome,price) $=$ value(outcome) $-$ price.
  - budgets:
    utility(outcome,price)
    $$= \begin{cases} \infty & \text{if price} > \text{budget} \\ \text{value(outcome) - price} & \text{otherwise} \end{cases}$$
  - etc.
- Approximation algorithms are ok.

Economic Optimization

Economic

Optimization

truthful

fair prices

Algorithmic

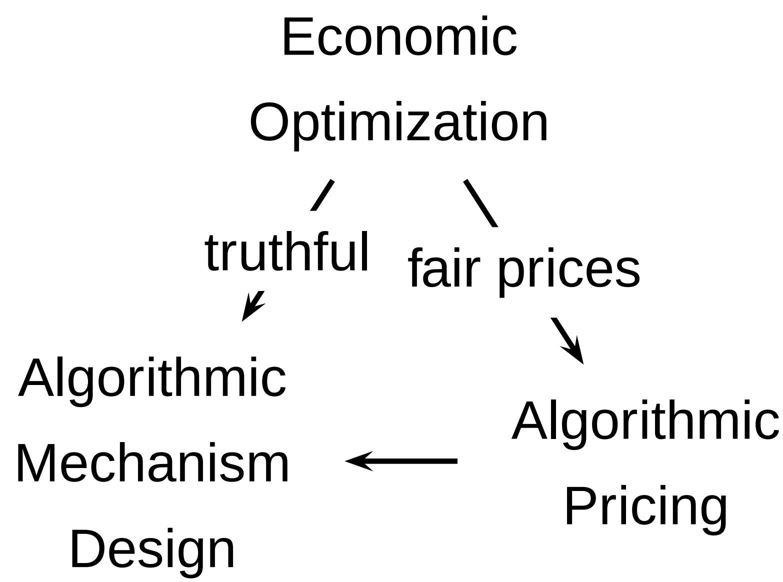
Mechanism

Design

Algorithmic

Pricing

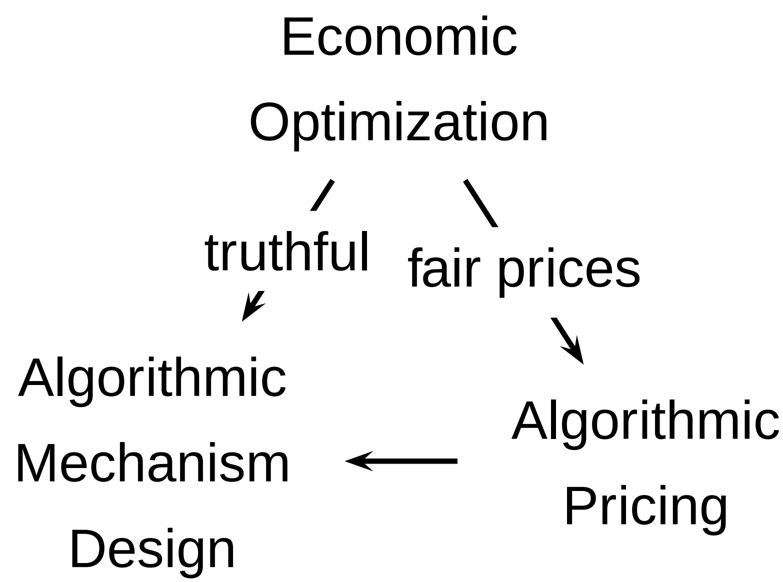
Economic Optimization



Conclusions:

- For additive objectives and “small” agents, random sampling reduces mechanism design to pricing.

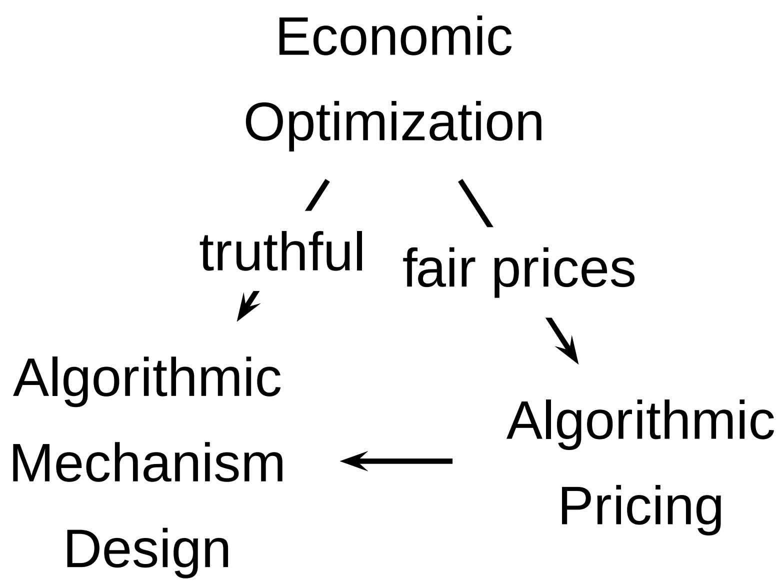
Economic Optimization



Conclusions:

- For additive objectives and “small” agents, random sampling reduces mechanism design to pricing.
- **Open:** algorithmic pricing.

Economic Optimization



Conclusions:

- For additive objectives and “small” agents, random sampling reduces mechanism design to pricing.
- **Open:** algorithmic pricing.
  (New direction: limited supply, welfare maximization.)

Economic Optimization

Economic Optimization

truthful ↙ ↘ fair prices

Algorithmic Mechanism Design ← Algorithmic Pricing

Conclusions:

- For additive objectives and “small” agents, random sampling reduces mechanism design to pricing.
- **Open:** algorithmic pricing.
  (New direction: limited supply, welfare maximization.)
- **Open:** non-linear objectives
  (e.g., makespan or non-additive costs).