Learning strong-substitutes demand correspondences

Edwin Lock
joint work with Paul W. Goldberg and Francisco Marmolejo
Department of Computer Science, University of Oxford

7 April, 2020

Introduction

Auctions - demand correspondences

Consider an auction setting with goods $\{1, \ldots, n\}$, where goods are available in multiple items.

Bidders need to communicate their demand to the auctioneer, in some fashion.

Bidders have a **demand correspondence** mapping price vectors $p \in \mathbb{R}^n$ to bundles demanded at those prices.

$$D(\boldsymbol{p}) := \{\text{bundles demanded at } \boldsymbol{p}\}$$

Quasi-linear demand correspondences

Classic definition

The **valuation function** v assigns a value to each bundle $\mathbf{x} \in \mathbb{Z}^n$.

The demand correspondence is defined as

$$D_v(\mathbf{p}) := \arg\max_{\mathbf{x} \in \mathbb{Z}^n} (v(\mathbf{x}) - \mathbf{p}.\mathbf{x}).$$

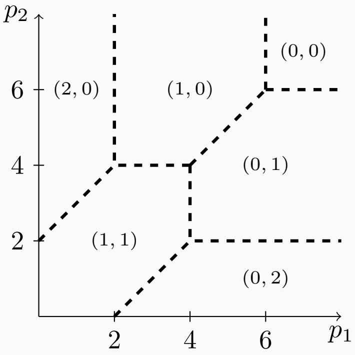
(We assume quasi-linear utilities $v(\mathbf{x}) - \mathbf{p}.\mathbf{x}$.)

Geometric approach (uses tropical geometry)

Theorem (Baldwin and Klemperer, 2019)
For every integral valuation, the demand correspondence $D_v(\mathbf{p})$ partitions price-space into an weighted integral polyhedral complex.

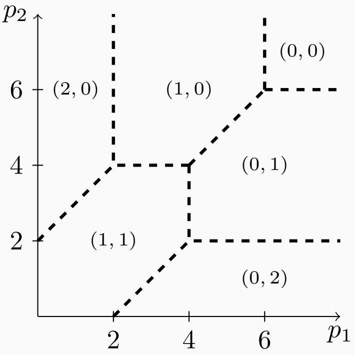
Strong-substitutes demand

Example demand with two goods



Strong-substitutes demand

Example demand with two goods



Theorem (Baldwin and Klemperer, 2019)

*A demand correspondence is **strong substitutes** iff its facets are normal to* $\mathbf{e}^i$ *or* $\mathbf{e}^i - \mathbf{e}^j$ *for some* $i, j \in \{1, \ldots, n\}$.

Representing a demand correspondence

How to represent demand?

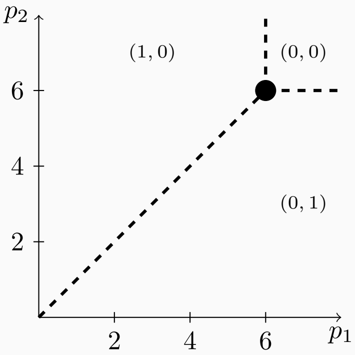
- Valuation
- Facets
- Vertices

The Product-Mix Auction [Klemperer 2009] introduces a **dot-bid language**.

Finite set of special (weighted) points in price space that define demand correspondence.

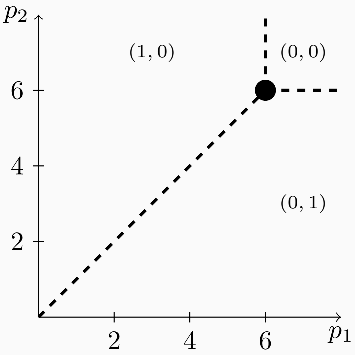
Dot bid language

Each bidder submits a list of **dot bids** $b \in \mathbb{Z}^n$ with weights $w_b = \pm 1$.



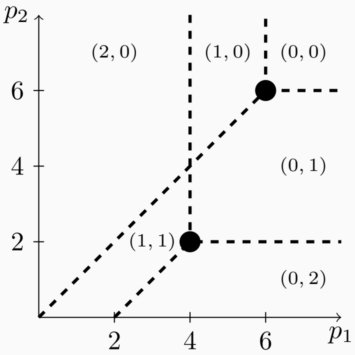
Dot bid language

Each bidder submits a list of **dot bids** $b \in \mathbb{Z}^n$ with weights $w_b = \pm 1$.



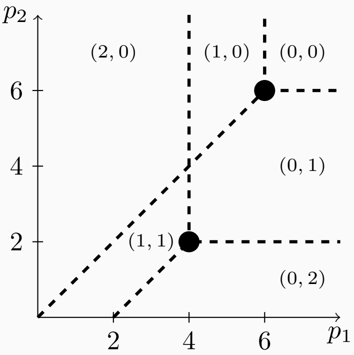
Dot bid language

Each bidder submits a list of **dot bids** $b \in \mathbb{Z}^n$ with weights $w_b = \pm 1$.



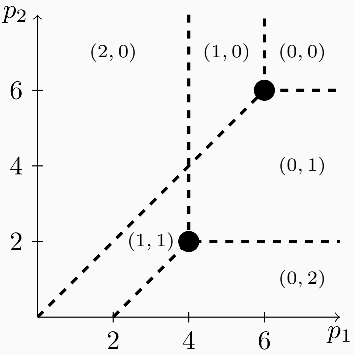
Dot bid language

Each bidder submits a list of **dot bids** $b \in \mathbb{Z}^n$ with weights $w_b = \pm 1$.

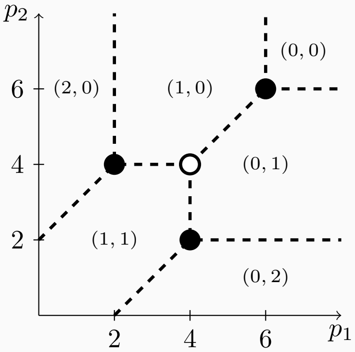


Dot bid language

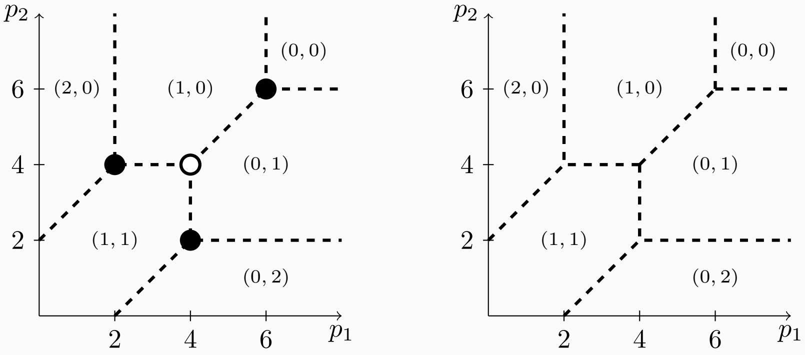
Each bidder submits a list of **dot bids** $b \in \mathbb{Z}^n$ with weights $w_b = \pm 1$.



Dot bid example



Dot bid example



Theorem (Baldwin and Klemperer, in preparation)

Every strong-substitutes demand correspondence can be expressed uniquely as a list of positive and negative dot bids.

The converse does not hold!

Learning dot-bid lists

Constructing dot bids

Suppose a bidder wishes to participate in the product-mix auction.

- In high dimensions (=many goods), it is not clear how to construct the list of dot bids ad hoc.
- A bidder may not have full information about their valuation or demand correspondence (in geometric or algebraic terms).

Constructing dot bids

Suppose a bidder wishes to participate in the product-mix auction.

- In high dimensions (=many goods), it is not clear how to construct the list of dot bids ad hoc.
- A bidder may not have full information about their valuation or demand correspondence (in geometric or algebraic terms).

Constructing dot bids

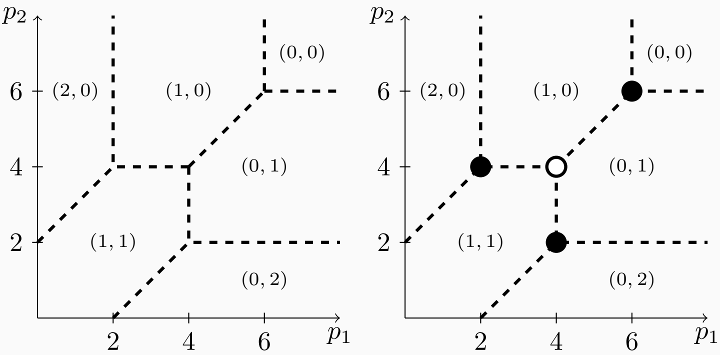
Suppose a bidder wishes to participate in the product-mix auction.

- In high dimensions (=many goods), it is not clear how to construct the list of dot bids ad hoc.
- A bidder may not have full information about their valuation or demand correspondence (in geometric or algebraic terms).

Challenge: Provide an algorithm that outputs the list of dot bids corresponding to a bidder's strong-substitutes demand.

We assume a **demand oracle** $\delta(\boldsymbol{p})$, which returns a bundle demanded at prices $\boldsymbol{p}$.

Constructing dot bids



Output: three positive bids at $(2, 4)$, $(4, 2)$ and $(6, 6)$, one negative bid at $(4, 4)$.

Our results

Challenge: Provide an algorithm that determines the list of dot bids corresponding to a bidder's strong-substitutes demand using access only to $\delta(\cdot)$.

Measure **query complexity**, in terms of size of output: number of goods n, number of bids B, and $\log M$, where $M = \max_{\boldsymbol{b}} \|\boldsymbol{b}\|_\infty$.

Our results

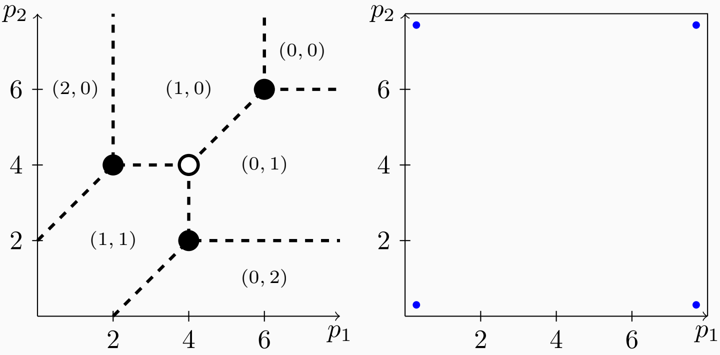
Challenge: Provide an algorithm that determines the list of dot bids corresponding to a bidder's strong-substitutes demand using access only to $\delta(\cdot)$.

Measure **query complexity**, in terms of size of output: number of goods n, number of bids B, and $\log M$, where $M = \max_{\boldsymbol{b}} \|\boldsymbol{b}\|_\infty$.

Main results

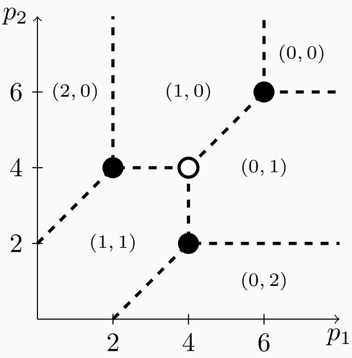
- A linear-time algorithm for learning bid lists in the case that all bids are positive (not discussed today).
- An exponential algorithm for the general case where bids can be positive and negative.
- A super-polynomial lower bound on the query complexity in the general case.

The hyperplane-finding algorithm



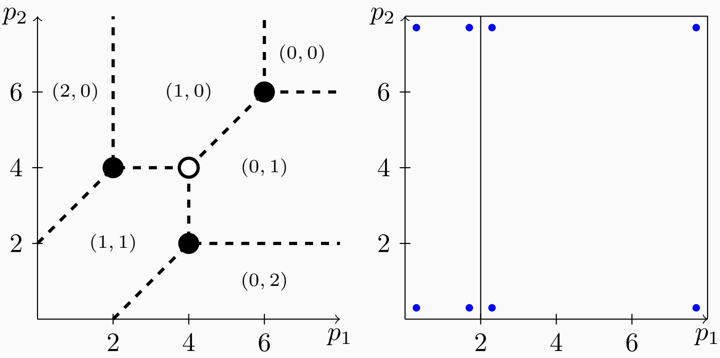
Stage 1: Learn all hyperplanes that contain a facet.

The hyperplane-finding algorithm



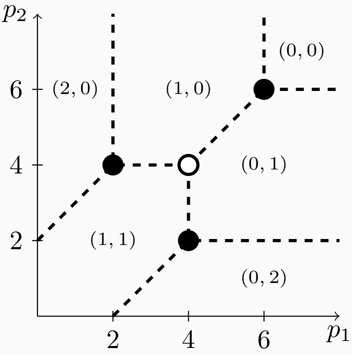
Stage 1: Learn all hyperplanes that contain a facet.

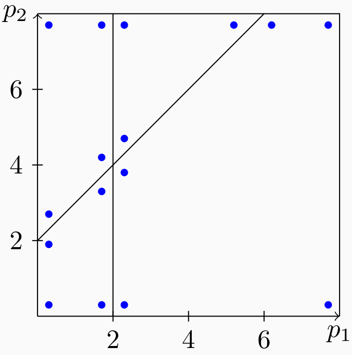
The hyperplane-finding algorithm



Stage 1: Learn all hyperplanes that contain a facet.

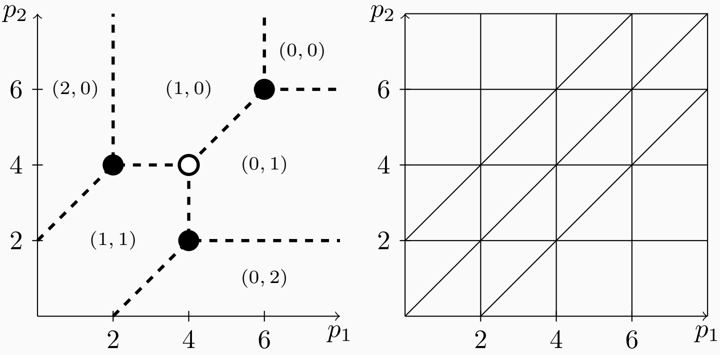
The hyperplane-finding algorithm





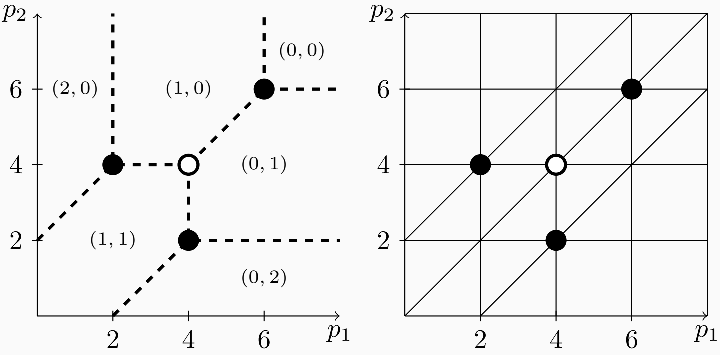
Stage 1: Learn all hyperplanes that contain a facet.

The hyperplane-finding algorithm



Stage 1: Learn all hyperplanes that contain a facet.

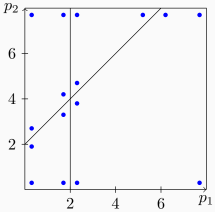
The hyperplane-finding algorithm



Stage 1: Learn all hyperplanes that contain a facet.

Stage 2: Check all intersections of *n* hyperplanes for the presence of a bid.

Query complexity



The hyperplane-finding algorithm has worst-case query complexity

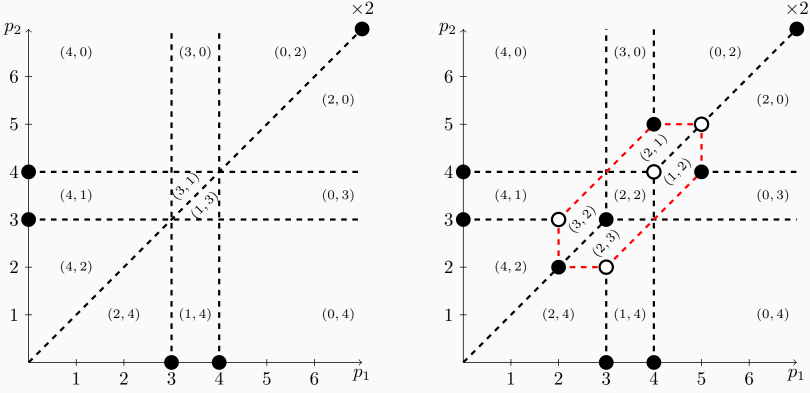
$$\underbrace{\binom{B\binom{n}{2}}{n} 2^n n!}_{\text{cost of intersection queries}} + \underbrace{O\left(B\binom{n}{2}\log M\right)}_{\text{cost of binary searching}}.$$

For n constant, this is

$$O\left(B^n + B\log(M)\right).$$

Island gadget lower bound

We introduce an **island gadget**.



The island gadget requires 2^{n+1} bids in n dimensions.

Island gadget lower bound

An **adversary** plays a game in which he places the gadget at one of $(M/4)^n$ candidate locations.

As a gadget only changes demand locally, a **player** must query inside the gadget to detect its presence.

Using this idea, it can be shown that

$$\Omega\left(\left(\frac{B - 2^{n+1}}{8n^2}\right)^n\right)$$

queries are required to learn the location of the gadget.

For n constant, this is

$$\Omega(B^n).$$

Outlook

Outlook

Our algorithms are interesting beyond their immediate application to the product-mix auction.

- Dot bids are a natural way to represent demand correspondences.
- The size of the bid list may be a good measure of the complexity of a demand correspondence.

Thank you!

Thank you!

Questions or comments? Please feel free to email me at edwinlock@gmail.com.