University of Zurich[UZH]
Department of Informatics

European Research Council
Established by the European Commission

Machine Learning-powered Iterative Combinatorial Auctions

Sven Seuken, University of Zurich

Preliminary version of this paper has previously appeared as:

Combinatorial Auctions via Machine Learning-based Preference Elicitation ***(IJCAI-ECAI 2018)***.

Joint work with: Gianluca Brero (University of Zurich) and Benjamin Lubin (Boston University)

October 18, 2019

NBER Market Design Working Group Meeting

Spectrum Auctions

- Governments are auctioning off multiple indivisible **licenses** (4G, 5G) among mobile network operators
- Bidders have value for **bundles of licenses**
- Licenses can be **substitutes** as well as **complements**

1. $80M on British Columbia
2. $60M on Alberta
3. $200M on British Columbia + Alberta

→ Direct revelation mechanisms (e.g., VCG) are infeasible
→ Need a mechanism with smart preference elicitation

Example: 2014 Canadian Spectrum Auction

- 10 bidders
- 98 different licenses
- Spread across 14 regions

→ 2^{98} bundles of licenses!

Iterative VCG Mechanisms (Mishra & Parkes'07; de Vries et al.'07)

Features:

– Interact with bidders over multiple rounds

– Elicit "enough" information to implement VCG outcome

– Straightforward truthful bidding is ex-post Nash equilibrium

However: Impossibility result by Nisan and Segal'06:

– To guarantee efficiency, we need exponential communication in the worst case

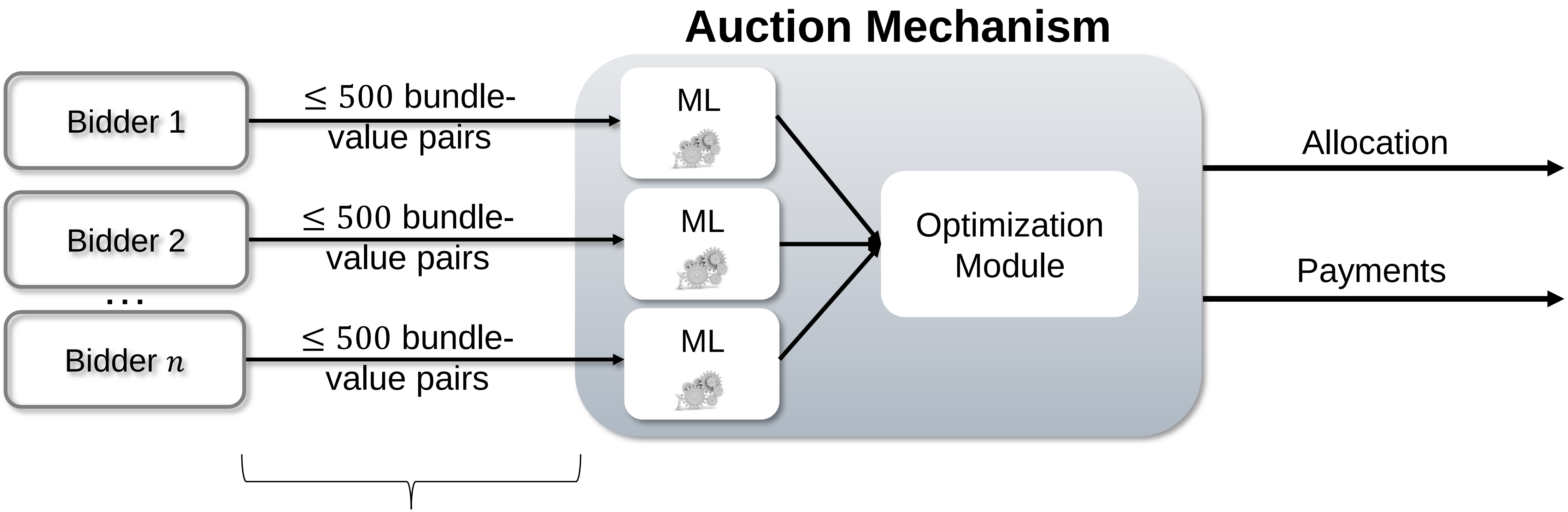
→ Practical auction designs (in domains with general valuations) cannot provide efficiency guarantees! → need to limit the amount of information exchanged

Combinatorial Clock Auction (CCA) (Ausubel, Cramton, Milgrom, 2006)

- Practical auction design:
  - Used in Switzerland, UK, Australia, **Canada**, etc. → more than $20 Billion in revenue
  - Informally: combines an "**ascending-price** phase" followed by a "**combinatorial sealed-bid** phase"
- Design features (that limit the amount of information exchanged):
  - Linear prices in the clock phase
  - Discrete price updates to keep the number of rounds small
  - At most 500 bids in the supplementary round
- **Inefficiencies of the CCA:**
  - Lab experiments → efficiencies of 89%-96% (Scheffel et al., 2013; Bichler et al., 2014)

1%-2% efficiency loss → can be ~$100 Million of welfare losses per auction!

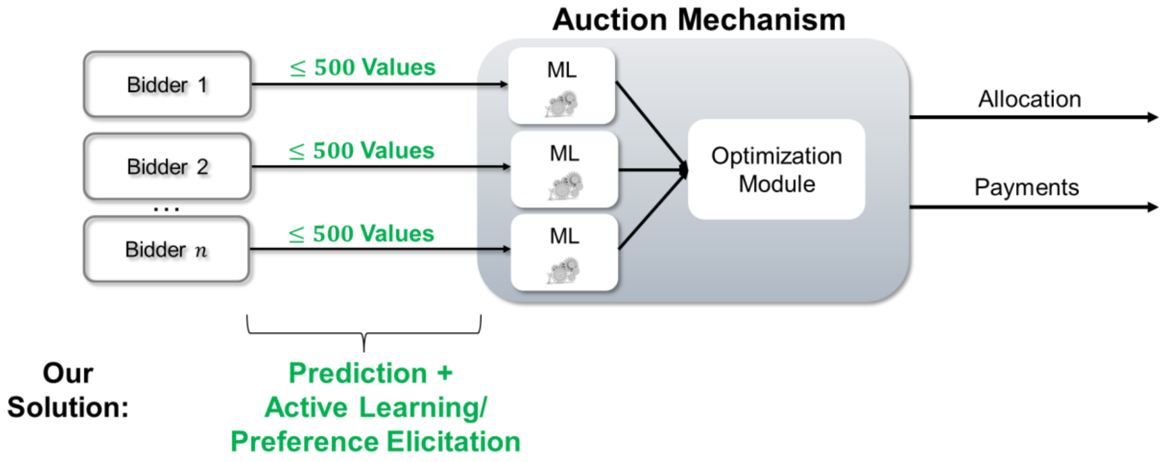
This Paper: A Machine Learning-powered Iterative Combinatorial Auction



Our Approach:

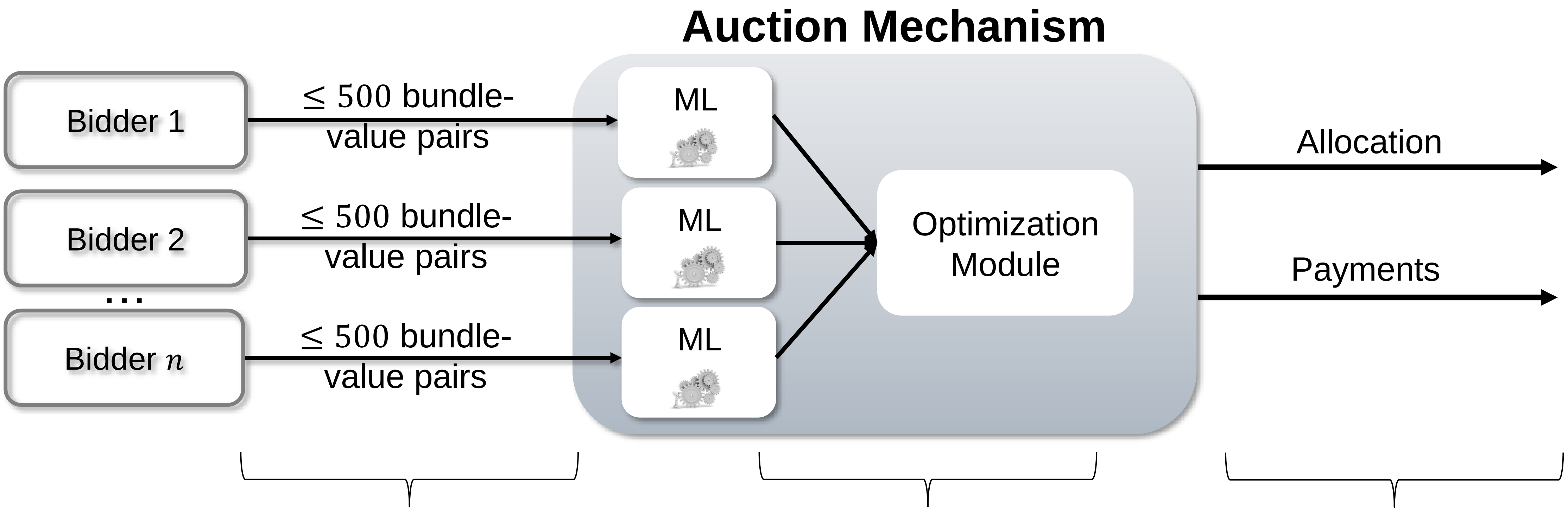
Prediction + Active Learning/ Preference Elicitation

What do I mean by "learning" or "prediction"?



- Bidders report (bundle, value)-pairs. For example:
  - (A, \$1); (B, \$2); (C, \$3); (AB, \$5)
- ML algorithm predicts values for all bundles in bundle space: e.g., (ABC, ?)
- **For now, think**: linear regression, with one coefficient per item
  - $\tilde{v}_i(x) = w_i \cdot x$
  - Example: $\tilde{v}_i(ABC) = w_A + w_B + w_C$ (Note: cannot capture complements or substitutes!)

This Paper: A Machine Learning-powered Iterative Combinatorial Auction



Our Approach:

Prediction + Active Learning/ Preference Elicitation

Optimization (with ML predictions)

Good Incentives via Pseudo-VCG Payments

Related work: Combining ML and Mechanism Design

- Early connections between “ML queries” and “auction queries”
  - Lahaie & Parkes (2004); Blum et al. (2004)

- “Learning clearing prices” in iterative CAs to achieve a small number of rounds
    - Lahaie (2011); Abernethy et al. (2016); Brero and Lahaie (2018); Brero, Lahaie, and Seuken (2019)

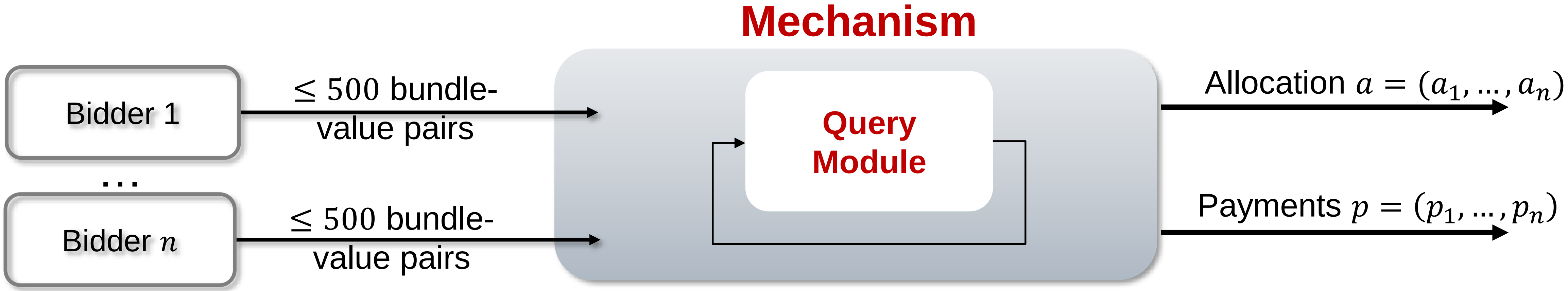
- Using ML to design better mechanisms (in the sense of “automated mechanism design”)
  - Dütting et al. (2015); Dütting et al. (2019); Narasimhan et al. (2016); Feng et al. (2018)

This work: integrating the ML algorithm *into* the CA and learning the bidders’ value functions

Outline

1. Motivation: Preference Elicitation in Combinatorial Spectrum Auctions
2. Our Machine Learning-powered Mechanism
3. Theoretical Analysis
4. Instantiating the ML Algorithm + Optimization Module
5. Experiments I: Choosing the best ML Algorithm
6. Experiments II: Comparing our mechanism against the CCA
7. Conclusion

Our Machine Learning-powered ICA – High Level View



- Component #1: Query Module
- Component #2: The Mechanism
- Goal: collect the 500 best bundle-value reports from each bidder to maximize empirical efficiency at the end
- Final allocation: Take all elicited values and solve the **winner determination problem (WDP)** [IP → CPLEX]

$$a^* = argmax_a \sum_i \hat{v}_i(a_i)$$

$$\text{s.t.} \quad \sum_i a_{ij} \leq 1 \ \forall j \in [m] \qquad a_{ij} \in \{0,1\} \ \forall i, j$$

The Machine Learning-powered Query Module – Schematic View

$\tilde{v}_i$= inferred value function for bidder i

s_i= set of bundle-value pairs from bidder i

q_i= next query for bidder i

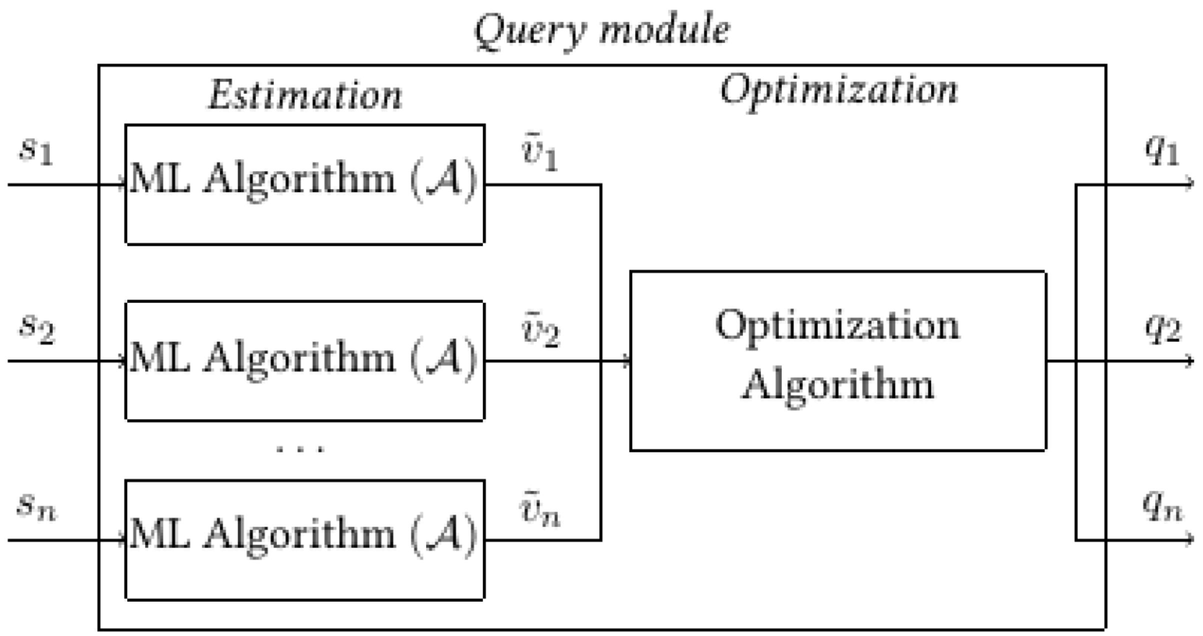


Figure 1: Schematic representation of how the query module works.

The Machine Learning-powered Query Module – Details

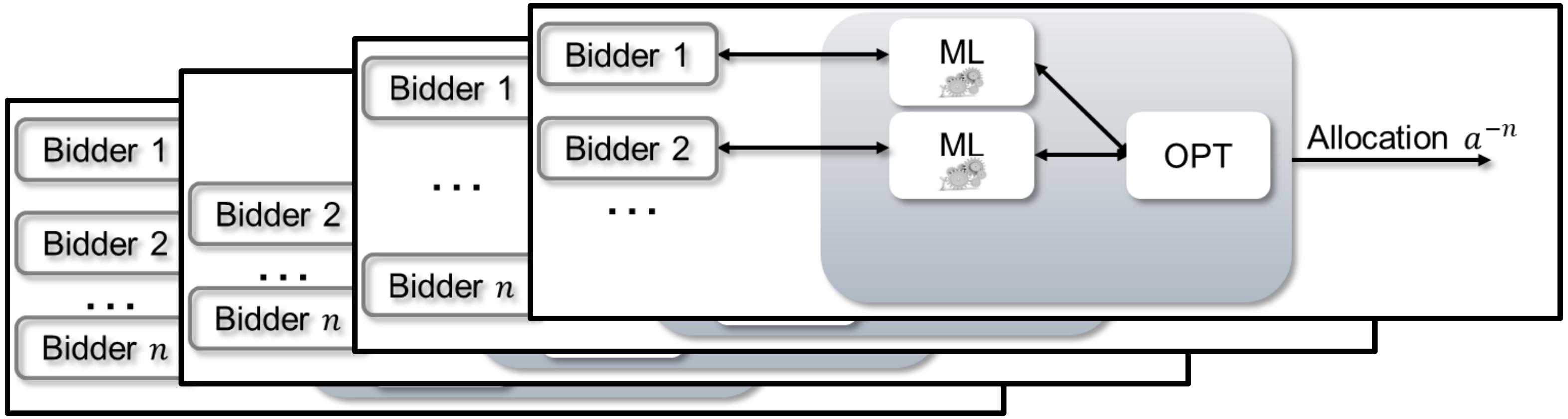
Algorithm 1: Machine Learning-powered Query Module

function NextQuery($\mathcal{A}, s$);
Inputs: ML algorithm $\mathcal{A}$; Vector of sets of bundle-value pairs $s = (s_1, ..., s_k)$);
foreach bidder $i \in [k]$ **do**
  $\tilde{v}_i = \mathcal{A}(s_i)$; **Estimation Step:** infer valuation for each bidder using ML algorithm
end
Determine $\tilde{a} \in \arg\max_{a \in \mathcal{F}} \sum_{i \in [k]} \tilde{v}_i(a)$; **Optimization Step** (based on inferred valuations)
foreach bidder $i \in [k]$ **do**
  if $\tilde{a}_i \notin s_i$ **then**
    $q_i = \tilde{a}_i$;
  else
    $\mathcal{F}_i = \{a \in \mathcal{F} : \forall x \in s_i,\ a_i \neq x\}$;
    Determine $\tilde{a}' \in \arg\max_{a \in \mathcal{F}_i} \sum_{i \in [k]} \tilde{v}_i(a)$; **Optimization Step** (with restrictions)
    $q_i = \tilde{a}'_i$;
  end
end
Output vector of queries $q = (q_1, ..., q_k)$;

The Pseudo-VCG Machine Learning-based (PVML) Mechanism

Two main design features:

1. Allow bidders to "push" bundle-value pairs in an initial round of the auction (e.g., 50-100)
2. Charge "VCG-style" payments at the end, by eliciting bundle-value pairs separately in:
   1. The "main economy" (with all n bidders)
   2. In each "marginal economy" of bidder i (where bidder i is excluded from the auction)



The PVML Mechanism – Details

Algorithm 2: Pseudo-VCG Machine Learning-based (PVML) Mechanism

Parameters: ML algorithm $\mathcal{A}$; maximum # of queries per bidder $Q_{\max}$; # of initial queries $Q_0 \leq Q_{\max}$;

1 Each bidder i submits up to $Q_i^{push} \leq Q_0$ self-chosen bundle-value pairs s_i^0;
2 Ask each bidder i to report his value for $Q_0 - Q_i^{push}$ randomly chosen bundles and add them to s_i^0;
3 Let $s^0 = (s_1^0, ..., s_n^0)$ denote the initial reports for the main economy;
4 For each bidder i, let $s^{0,(-i)} = (s_1^0, ..., s_{i-1}^0, s_{i+1}^0, ..., s_n^0)$ be the initial reports for i's marginal economy;
5 Initialize round counter: $t = 0$;
6 **while** $\max_i |s_i^t| \leq Q_{\max} - n$ **do**
7 $\quad t = t + 1$;
8 $\quad$ Generate queries for the main economy: *NextQuery*$(\mathcal{A}, s^{t-1})$;
9 $\quad$ Generate queries for each bidder i's marginal economy: *NextQuery*$(\mathcal{A}, s^{t-1,(-i)})$;
10 $\quad$ Send generated queries to bidders and ask for corresponding values;
11 $\quad$ Let s' denote all reported bundle-value pairs obtained in Step 10 and let $s^t = s^{t-1} \cup s'$;
12 $\quad$ Let $s'^{(-i)}$ denote the reported bundle-value pairs obtained in Step 10 for bidder i's marginal economy and let $s^{t,(-i)} = s^{t-1,(-i)} \cup s'^{(-i)}$;
13 **end**
14 Determine allocation $a^{pvml} = a^*_{\hat{v}^\star}$, where $\hat{v}^\star = \hat{v}_{s^t}$;
15 Charge each bidder i payment

$$p_i^{pvml} = \sum_{j \neq i} \hat{v}_j^{(-i)}(a^{(-i)}) - \sum_{j \neq i} \hat{v}_j^\star(a^{pvml}), \qquad \text{where } \hat{v}^{(-i)} = \hat{v}_{s^{t,(-i)}} \text{ and } a^{(-i)} = a^*_{\hat{v}^{(-i)}}; \tag{3}$$

16 Output allocation a^{pvml} and payments p^{pvml};

Theoretical Analysis

1. **Relationship between learning error and performance of PVML**
2. **Good Incentives in Practice**
3. Individual Rationality
4. No-deficit

Bounding the Efficiency Loss

- Learning error in bundle x for bidder i: $|\tilde{v}_i(x) - v_i(x)|$

Proposition 1. *Let $\tilde{v}$ be an inferred valuation profile. Let $a^*_{\tilde{v}}$ be an efficient allocation w.r.t. to $\tilde{v}$, and let a^*_v be an efficient allocation w.r.t. the true valuation profile. Assume that the learning errors in the bundles of these two allocations are bounded as follows: for each bidder i, $|\tilde{v}_i(a^*_{\tilde{v}}) - v_i(a^*_{\tilde{v}})| \leq \delta_1$ and $|\tilde{v}_i(a^*_v) - v_i(a^*_v)| \leq \delta_2$, for $\delta_1, \delta_2 \in \mathbb{R}$. Then the following bound on the efficiency loss in $a^*_{\tilde{v}}$ holds:*

$$\frac{V(a^*_v) - V(a^*_{\tilde{v}})}{V(a^*_v)} \leq \frac{n(\delta_1 + \delta_2)}{V(a^*_v)}. \qquad (4)$$

→ Provides motivation for the iterative design of the Query Module (reduce learning error)

Imputing Prices in PVML

- PVML does *not* use prices to communicate with bidders!
- But: we can *impute prices* to gain insight into how PVML “implicitly prices bundles” throughout the auction
- Let $\pi = (\pi_1, \ldots, \pi_n)$ be a general price function profile (allowing for non-anonymous bundle prices)

Definition 2 (Competitive equilibrium). *Given prices* π*, we define each bidder* i*'s demand set* d_i^π *as the set of bundles that maximize her utility at* π*:* $d_i^\pi = \arg\max_{x \in \mathcal{X}} v_i(x) - \pi_i(x)$*. Similarly, we can define the seller's supply set* s^π *as the set of allocations that are most profitable at* π*:* $s^\pi = \arg\max_{a \in \mathcal{F}} \sum_i \pi_i(a_i)$*. We say that prices* π *and allocation* a *are in competitive equilibrium if* $a \in s^\pi$ *and, for each bidder* i*,* $a_i \in d_i^\pi$*.*

Approximate Competitive Equilibrium Prices in PVML

Consider imputed prices $\pi = \tilde{v}$

Proposition 2. *Let $\tilde{v}$ be an inferred valuation profile and a_v^* be an efficient allocation. Assume that the learning errors are bounded as follows: for each bidder i, $\max_{x \in \mathcal{X}} |\tilde{v}_i(x) - v_i(x)| \leq \delta_1$ and $|\tilde{v}_i(a_v^*) - v_i(a_v^*)| \leq \delta_2$. Then, we need to inject at most $n(\delta_1 + \delta_2)$ into the market to induce the bidders and the seller to trade the allocation a_v^* at prices $\pi = \tilde{v}$, i.e., $\tilde{v}$ is a $n(\delta_1 + \delta_2)$-approximate competitive equilibrium price profile.*

- Proposition 2 provides a measure of the quality of the prices $\pi = \tilde{v}$
- Implicit price structure depends on ML algorithm used → prices will, in general, be non-anonymous bundle prices → thus, more powerful than anonymous linear prices
- Connection to Lahaie & Parkes'04
  - Propose an elicitation algorithm similar to ours; guarantees finding a CE
  - However, in each round, they communicate (exponentially-sized) ask prices to bidders

Incentives: Social Welfare Alignment and "Bidder Push"

- **PVML is manipulable** (dynamic strategies and no efficiency guarantees)
- **Theorem: If other bidders are truthful, then PVM aligns incentives with efficiency**
- **Proof Sketch:** Utility of bidder i under PVM: $u_i = v_i(a^{pvm}) - p_i^{pvm}$

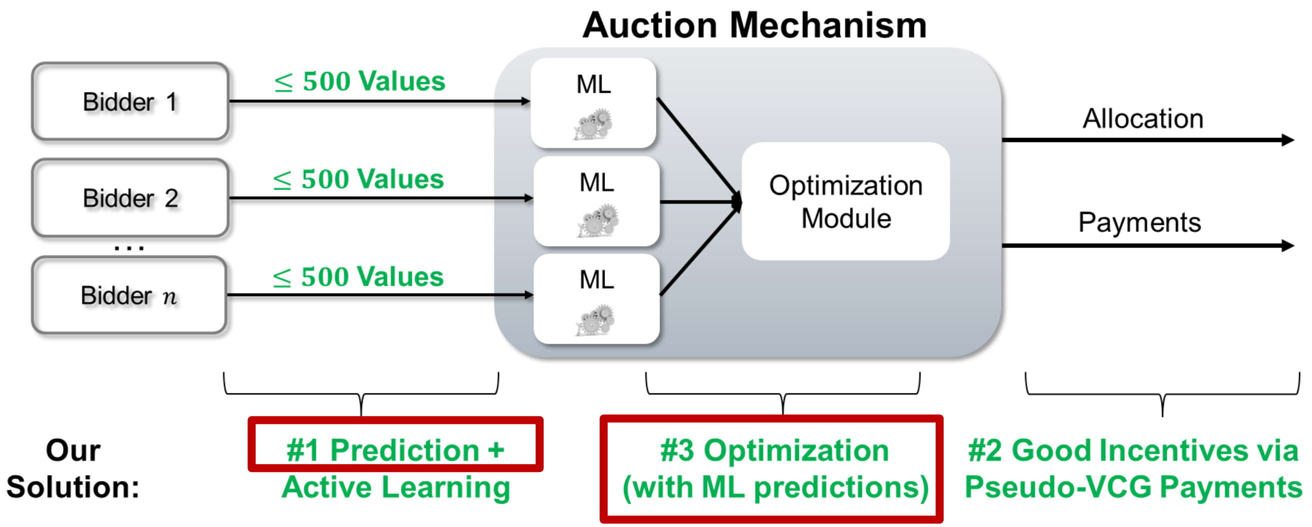
$$= \underbrace{v_i(a^{pvm}) + \sum_{j \neq i} \hat{v}_j(a^{pvm})}_{\text{Welfare w.r.t. bidder } i\text{'s true valuation}} - \underbrace{\sum_j \hat{v}_j(a^{-i})}_{\text{Independent of bidder } i\text{'s report}}$$

→ **If bidder finds a beneficial manipulation, this will maximize welfare w.r.t. to true values.**

→ **Good incentives in practice:** together with "bidder-push", this provides incentives to:

- (a) Push the bundles you believe will be part of an efficient allocation
- (b) Only submit truthful value reports

Which Machine Learning Algorithm to Use?



Need ML algorithm with two properties:

1. Good from economic perspective (predicting non-linear values) **and** works with small amount of data
2. Good from computational perspective (integrate ML into optimization and remain computationally feasible)

→ Start with **linear regression** (to explain the concept) and then move on to **SVRs with non-linear kernels**

Machine Learning: Linear Regression

- **Input:** ℓ reported bundle-value pairs $\{(x_1, v_1), (x_2, v_2), \ldots, (x_\ell, v_\ell)\}$
- **Goal:** predict value function $\tilde{v}_i(x)$
- **Standard linear regression:**
  - $\tilde{v}_i(x) = w_i \cdot x$, where $w_i \cdot x = \sum_j w_{ij} x_j$ [w_{ij} is bidder i's predicted value for item j]
  - →find coefficient vector w_i such that $\tilde{v}_i(x)$ is as accurate as possible on reported values:

$$\min \sum_{k=1}^{l} L(v_{ik}, w_i \cdot x_k)$$

  - In linear regression, we typically use the *squared loss function*: $L_2(y, \tilde{y}) = (y - \tilde{y})^2$
- **Regularized linear regression**: avoid overfitting → introduce a *regularization* term (min. magnitude of w_i)

$$\min \ ||w_i||^2 + C \sum_{k=1}^{l} L(v_{ik}, w_i \cdot x_k)$$

Winner Determination (using Linear Regression)

$$\max_{a} \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij}\, a_{ij}$$

s.t. $\sum_i a_{ij} \leq 1 \ \forall j \in [m]$ (feasibility constraint)

- $a_{ij} \in \{0,1\}$ are the decision variables (does bidder i get item j)
- w_{ij} are the learned coefficients from the linear regression (constants here)

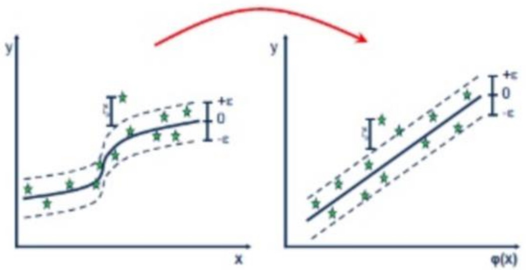
Computational difficulty:

- Winner determination is NP-hard
- This Integer Program (IP) has $n \cdot m$ Boolean variables and m constraints
- Using CPLEX (branch and bound) we can solve large instances (10 bidders, 98 items) in seconds

→ Limitation of linear regression-based approach: **cannot capture complements or substitutes!**

Support Vector Regression (SVR)

- **From linear to non-linear models:**
  - Linear model: $\tilde{v}_i(x) = w_i \cdot x$
  - Non-linear model: $\tilde{v}_i(x) = w_i \cdot \varphi(x)$
  - SVR: $\min \left|\left|w_i\right|\right|^2 + C \sum_{k=1}^{l} L_\varepsilon(v_{ik}, w_i \cdot \varphi(x_k))$ $[L_\varepsilon = \max\{|y - \tilde{y}| - \varepsilon, 0\}]$
  - Winner determination (primal): $\text{argmax}_a \sum_i w_i \, \varphi(a_i)$ (size depends on φ, i.e., number of features)
  - For low-dimensional feature spaces: easy to minimize w_i, but not for high-dimensional spaces



(do linear regression in feature space)

- **SVRs with non-linear kernels:**
  - Use the "kernel trick": find a $\kappa()$ such that $\varphi(x) \cdot \varphi(x') = \kappa(x, x')$
  - Predicted valuation: $\tilde{v}_i(x) = \sum_{k=1}^{\ell} \beta_{ik} \, \kappa(x_{ik}, x)$, where the x_{ik} are bundles evaluated by bidder i
  - Winner determination (dual): $\text{argmax}_a \sum_i \sum_{k=1}^{l} \beta_{ik} \, \kappa(x_{ik}, a_i)$ (size depends on # of reported values)

  → Need to choose a "good" kernel function κ!

Choosing a Kernel Function

Linear Kernel
$\kappa(x, x') = x \cdot x'$

Quadratic Kernel
$\kappa(x, x') = (x \cdot x') + \lambda(x \cdot x')^2$

Exponential Kernel
$\kappa(x, x') = \exp(x \cdot x')$

Gaussian Kernel
$\kappa(x, x') = \exp(-||x - x'||^2)$

Captures non-additivity
(complements and substitutes)

Winner Determination Problem (using the Dual) with Quadratic Kernel

$$\max_{a} \sum_{i} \sum_{k=1}^{l} \beta_{ik} \left(x_{ik} a_i\right) + \gamma \beta_{ik} \left(x_{ik} a_i\right)^2$$

s.t. $\sum_i a_{ij} \leq 1 \;\forall j \in [m]$ (feasibility constraint)

- $a_{ij} \in \{0,1\}$ are the decision variables (does bidder i get item j)
- β_{ik} are the learned coefficients (dual variables) from the SVR (constant here)
- x_{ik} is bundle k reported by bidder i (support vector from the dual of the SVR)
- γ is the Kernel parameter

Computational difficulty:

- This is a Quadratic Integer Program (QIP)
- CPLEX can solve large instances (10 bidders, 98 items) within 1h within a relative MIP gap of <= 2%

Experiments: Measure Efficiency of Mechanisms

- 2014 Canadian auction is only one data point!
- → **We use a data generator: "SATS: A Universal Spectrum Auction Test Suite"** (Weiss et al. '17)
  - On demand, SATS can create thousand of (random) spectrum auction instances
  - SATS has access to all bidders' value functions → we can compute the efficient allocation
  - We can use the value function to answer value queries and demand queries

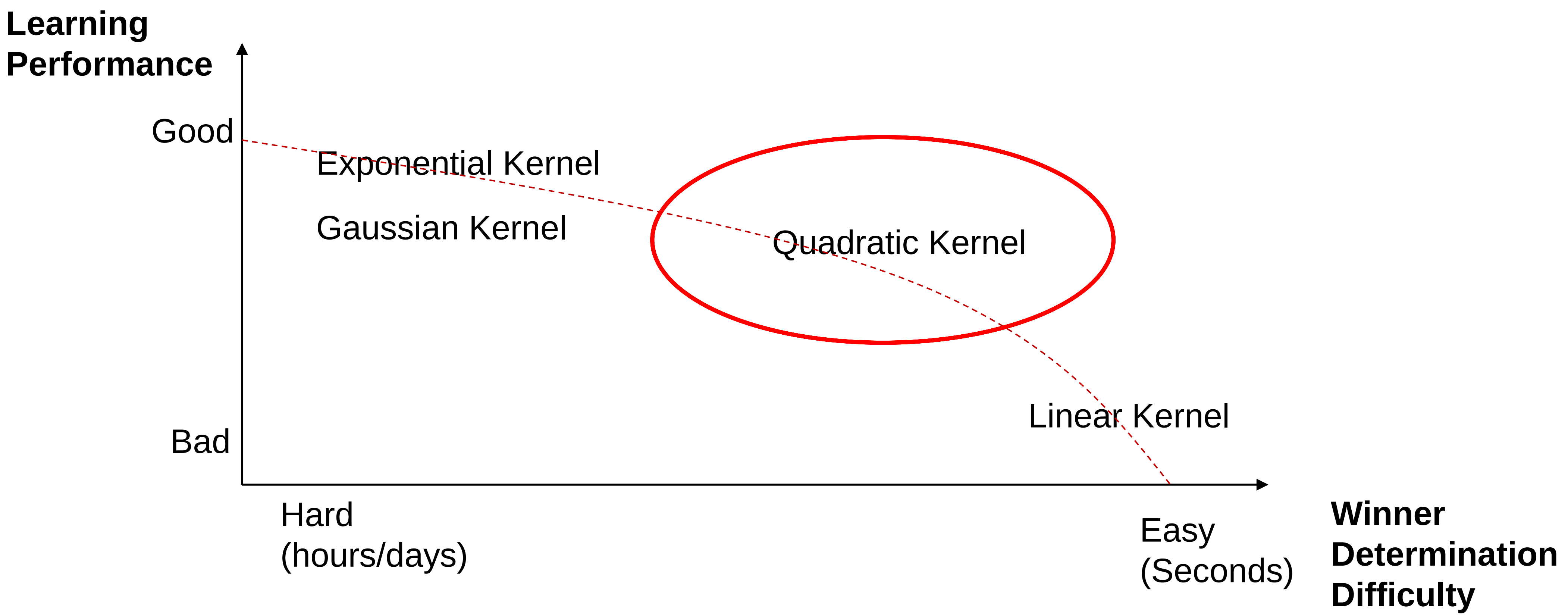
- SATS contains many spectrum value models, we tested on three:
  1. GSVM Model, 18 items, 7 bidders (Goeree and Holt, 2008)
  2. LSVM Model, 18 items, 6 bidders (Scheffel et al., 2012)
  3. 2014 Canadian Auction Model, 98 items, 10 bidders (Weiss et al., 2017)

Optimizing the ML Algorithm (= Choosing the Best Kernel)

Kernel	ϵ	Efficiency			Learning Error			WD Solve Time			Optimality Gap		
		100	200	500	100	200	500	100	200	500	100	200	500
Exponential	0	83.0%	83.5%	69.8%	15.68	13.86	11.66	60.00s	60.00s	60.00s	2.40	7.46	109.35
Exponential	16	83.3%	83.5%	83.6%	18.58	16.21	13.86	20.04s	59.76s	60.00s	0.06	0.89	5.80
Exponential	32	83.2%	83.7%	83.7%	24.07	22.28	20.82	1.39s	10.11s	60.00s	0.00	0.01	1.22
Gaussian	0	66.3%	56.3%	-	17.17	14.70	-	60.00s	60.00s	-	6.20	23.46	-
Gaussian	32	76.2%	78.1%	78.7%	27.15	24.53	21.88	58.47s	60.00s	60.00s	0.34	1.41	4.89
Gaussian	64	78.1%	81.8%	82.1%	38.32	36.24	34.44	11.79s	35.21s	59.58s	0.00	0.02	0.43

Kernel	Efficiency			Learning Error			WD Solve Time			Optimality Gap		
	100	200	500	100	200	500	100	200	500	100	200	500
Linear	72.9%	76.0%	74.8%	22.83	21.36	20.58	0.00s	0.00s	0.01s	0.00	0.00	0.00
Quadratic	88.8%	92.6%	93.2%	16.83	14.59	12.62	0.08s	0.16s	0.21s	0.00	0.00	0.00
Exponential	83.2%	83.7%	83.7%	24.07	22.28	20.82	1.39s	10.11s	60.00s	0.00	0.01	1.22
Gaussian	78.1%	81.8%	82.1%	38.32	36.24	34.44	11.79s	35.21s	59.58s	0.00	0.02	0.43

The Quadratic Kernel lies on a Pareto Frontier of Learning Performance and Winner Determination Complexity (in our domains)



Comparing PVML against CCA – Experimental Set-up

PVML:

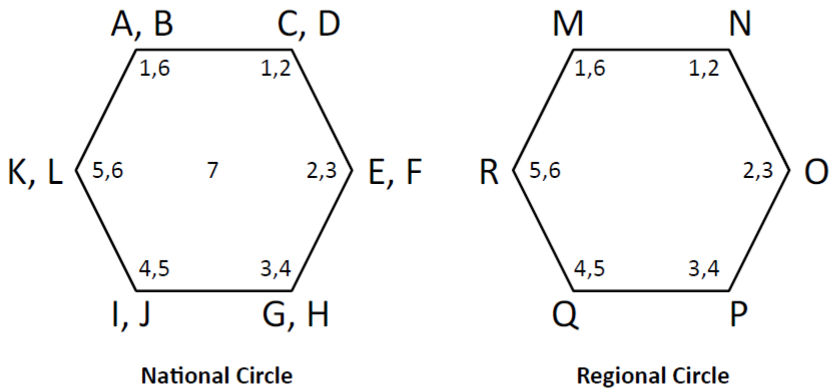
- Quadratic kernel
- Maximum number of queries = {100, 200, 500}
- Initial number of queries between 50 and 90 (here: chosen uniformly at random from the bundle space)

CCA:

- 5% price update rule in the clock phase (starting at low, but reasonable reserve prices)
- We simulate bidders who answer demand queries perfectly
- In the supplementary round, bidders submit {100, 200, 500} bids according to 3 different heuristics

Both mechanisms: simulate straightforward truthful bidding

Comparison of PVML vs. CCA – in the GSVM Domain (7 Bidders, 18 Goods)



Mechanism	Heuristic	Query Cap	Efficiency	Rounds
VCG			100.0%	1
CCA	Clock Bids		94.2%	118
	Clock Bids Raised		96.8%	118
	Profit Max	100	99.2%	118
	Profit Max	200	99.6%	118
	Profit Max	500	99.7%	118
PVML		100	100.0%	6
		200	100.0%	41
		500	100.0%	153

Comparison of PVML vs. CCA – in the LSVM Domain (6 Bidders, 18 Goods)

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q*	R

Domain: 18 items, 6 bidders
Value depends on “spatial proximity”

Mechanism	Heuristic	Query Cap	Efficiency	Rounds
VCG			100.0%	1
CCA	Clock Bids		81.4%	124
	Clock Bids Raised		90.9%	124
	Profit Max	100	99.4%	124
	Profit Max	200	99.8%	124
	Profit Max	500	99.9%	124
PVML		100	98.6%	13
		200	99.1%	37
		500	99.7%	113

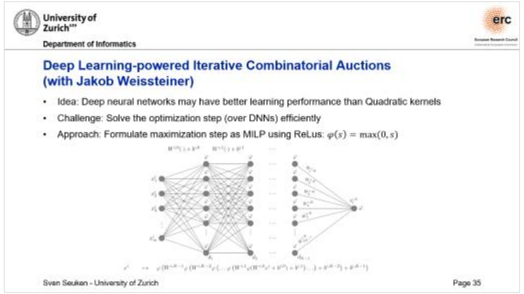
Comparison of PVML vs. CCA – MRVM Domain (10 Bidders, 98 Goods)

Mechanism	Heuristic	Query Cap	Efficiency	Rounds
VCG			100.0%	1
CCA	Clock Bids		93.0%	140
	Clock Bids Raised		93.2%	140
	Profit Max	100	92.0%	140
	Profit Max	200	92.1%	140
	Profit Max	500	92.4%	140
PVML		100	91.5%	13
		200	93.3%	25
		500	94.6%	56

Conclusion and Outlook

- **Design of an ML-powered Iterative Combinatorial Auction**
  1. Used ML to predict bidders' value functions
  2. Exploited properties of SVRs to find efficient allocation
  3. Used "bidder push" and "Pseudo-VCG" payments to induce good incentives
  4. Experimental results suggest better performance than CCA in large domains

- **Future/Ongoing Work:**
  1. Bidders report upper/lower bounds instead of exact values
  2. Other non-linear learning models (e.g., deep neural networks)

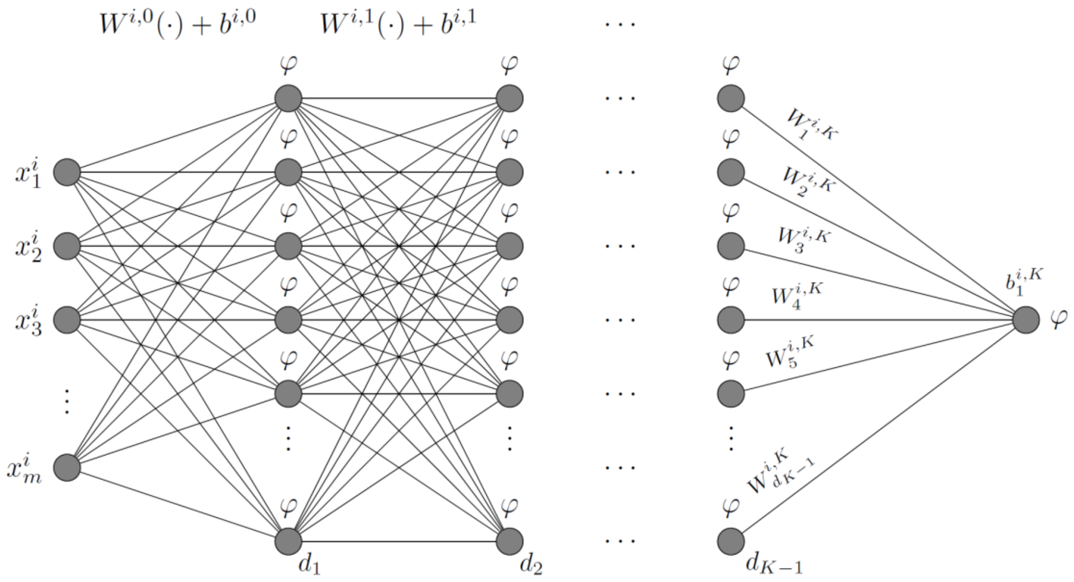


Thank you for your attention!

Backup

Deep Learning-powered Iterative Combinatorial Auctions (with Jakob Weissteiner)

- Idea: Deep neural networks may have better learning performance than Quadratic kernels
- Challenge: Solve the optimization step (over DNNs) efficiently
- Approach: Formulate maximization step as MILP using ReLus: $\varphi(s) = \max(0, s)$



$$x^i \quad \mapsto \quad \varphi\left(W^{i,K-1}\varphi\left(W^{i,K-2}\varphi\left(\ldots\varphi\left(W^{i,1}\varphi(W^{i,0}x^i + b^{i,0}) + b^{i,1}\right)\ldots\right) + b^{i,K-2}\right) + b^{i,K-1}\right)$$