Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Probably Approximately Efficient Combinatorial Auctions via Machine Learning

Gianluca Brero
Department of Informatics
University of Zurich
brero@ifi.uzh.ch

Benjamin Lubin
Information Systems Department
Boston University School of Management
blubin@bu.edu

Sven Seuken
Department of Informatics
University of Zurich
seuken@ifi.uzh.ch

Abstract

A well-known problem in combinatorial auctions (CAs) is that the value space grows exponentially in the number of goods, which often puts a large burden on the bidders and on the auctioneer. In this paper, we introduce a new design paradigm for CAs based on machine learning (ML). Bidders report their values (bids) to a proxy agent by answering a small number of value queries. The proxy agent then uses an ML algorithm to generalize from those bids to the whole value space, and the efficient allocation is computed based on the generalized valuations. We introduce the concept of *probably approximate efficiency (PAE)* to measure the efficiency of the new ML-based auctions, and we formally show how the generelizability of an ML algorithm relates to the efficiency loss incurred by the corresponding ML-based auction. To instantiate our paradigm, we use support vector regression (SVR) as our ML algorithm, which enables us to keep the winner determination problem of the CA tractable. Different parameters of the SVR algorithm allow us to trade off the expressiveness, economic efficiency, and computational efficiency of the CA. Finally, we demonstrate experimentally that, even with a small number of bids, our ML-based auctions are highly efficient with high probability.

1 Introduction

Combinatorial auctions (CAs) allow bidders to submit bids on bundles of items. They have found widespread applications in practice, for example for the procurement of industrial goods (Sandholm 2013), for selling TV-ad slots (Goetzendorf et al. 2015), and in government spectrum auctions (Cramton 2013). A motivation for allowing bidders to bid on bundles, instead of just on individual items, is that this avoids the "exposure problem" and can increase efficiency.

1.1 Preference Elicitation in CAs

Unfortunately, a major challenge when conducting CAs in practice is the fact that the bundle space grows exponentially in the number of items, which makes it impossible for the bidders to report their full value function for even medium-sized problems. The AI community has studied this problem from multiple angles, recognizing that careful preference elicitation is a formidable challenge in CAs. For example, Afriat (1967) and Lahaie (2010a) studied how to derive value functions that *rationalize observations* where bidders communicate their favorite bundles at given prices. Lahaie and Parkes (2004) proposed an *elicitation algorithm* using demand and value queries based on learning algorithms. However, this algorithm may require a very large amount of communication between the mechanism and the bidders (exponential in the number of items).

Blumrosen and Nisan (2009) showed that elicitation algorithms based on demand queries achieve the best possible approximation of the optimal allocation under polynomially-sized communication. However, in the worst case, these approximations can still be bad. Furthermore, all of these approaches based on demand queries do not consider the computational burden that demand queries impose on bidders (Blumrosen and Nisan 2009). Along the same lines, Scheffel, Ziegler, and Bichler (2012) showed that the price feedback provided by demand queries may provide a cognitive challenge for bidders, and may not be effective in coordinating them towards an efficient outcome.

1.2 Machine Learning-based Auctions

Our goal in this paper is to propose a radically new auction design paradigm which only elicits a small number of bids from the bidders and then uses a machine learning algorithm to *generalize* to the whole value space. We are willing to forego full efficiency to significantly lower the communication burden on the bidders.

There are numerous challenges that need to be addressed when using an ML algorithm inside a CA. The primary worry, of course, is what happens to economic efficiency when the generalized value function is wrong. To address this question, we introduce a new concept we call *Probably Approximate Efficiency (PAE)* as a useful relaxation of the standard efficiency notion from auction design. PAE enables us to capture the efficiency of ML-based auctions, by proving that they incur less than an ε loss in efficiency, with a probability $1 - \delta$. Readers familiar with the *Probably Approximately Correct (PAC)* concept from ML will notice the intended similarity to the PAC concept (Kearns and Vazirani 1994); however, while the two concepts are similar in spirit, they are quite different on a technical level.[1]

[1] While PAC uses generalization errors via an expectation over all points, we must use point-wise prediction intervals.

To instantiate our new paradigm, we use *support vector regression (SVR)* algorithms using carefully chosen kernels, as this gives us various desirable properties and good results in terms of efficiency: we show experimentally that our ML-based CAs are highly efficient with high probability. To minimize the efficiency loss as much as possible in practice, we propose an *iterative version* of our auction design, where the auction queries the bidders for their value of the allocated bundle. This allows the auction to correct the generalized valuation in the parts of the value space that is likely to be most relevant for determining the efficient allocation. Note that prior work has already considered kernels in research on combinatorial markets, e.g., for computing sufficiently expressive clearing prices (Lahaie 2009; 2010b; 2011; Abernethy, Lahaie, and Telgarsky 2016).

A second worry is the *expressiveness* of the auction, i.e., the question of whether bidders are able to express their values when their interaction with the auction is mediated by an ML algorithm. Indeed, not all choices of ML algorithms will allow the bidders to express their full value function. We study the expressiveness properties of different choices of kernels, and show that (for suitably chosen parameters), polynomial kernels are expressive for a restricted value space, while exponential and Gaussian kernels are fully expressive (for all value functions).

A third worry in the design of CAs is the complexity of the winner determination problem. If the bidders' value functions are communicated to the auction center via an ML algorithm, it is unclear how the WD problem should be solved. For some choices of an ML algorithm, the WD problem would become non-linear, which would prohibit an efficient solving of the problem. In this paper, we also provide a succinct integer programming (IP) formulation for the WD problem that arises from using SVR with the three types of kernels we employ. This enables us to solve reasonably-sized auction instances using standard optimization software.

2 Preliminaries

In a combinatorial auction (CA), there is a set M of m distinct, indivisible items, and a set N of n bidders. We let $s_i \in \{0,1\}^m$ denote a particular bundle under consideration by bidder i, with $s_{ij} = 1$ iff the bundle contains item j. Given a bundle s_i for every bidder, we let $s = (s_1, ..., s_n)$ denote the vector of those bundles.[2] Each bidder i has a *value function* v_i which, for every bundle s_i, defines i's value $v_i(s_i) \in \mathbb{R}_{\geq 0}$. Given a value function v_i for each bidder, we let $v = (v_1, ..., v_i, ..., v_n)$ denote the corresponding valuation profile.

The CA asks bidders for value reports on different bundles. Bidders may make non-truthful value reports. However, in this paper we do not study the incentive properties of the CA, and thus we do not distinguish between true and reported values.

We use $a = (a_1, ..., a_n)$ to denote an *allocation*, with a_i being the bundle that i gets allocated. An allocation a is *feasible* if $\forall j : \sum_i a_{ij} \leq 1$. We let $\mathbb{A}$ denote the set of feasible allocations. Given a feasible allocation $a \in \mathbb{A}$, we let $V(a) = \sum_i v_i(a_i)$ denote the *social welfare* of a.

[2] When used in the machine learning algorithm, this nested vector needs to be flattened into a $(n \cdot m)$-dimensional vector.

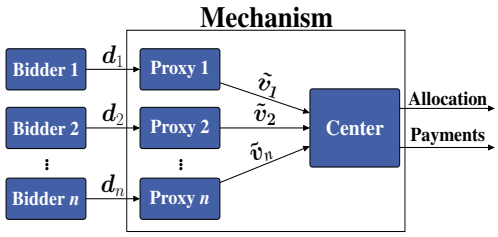

Figure 1: Schematic view of the ML-based Mechanism.

The winner determination problem (determining the welfare-maximizing allocation) is an NP-hard combinatorial optimization problem. We denote the solution to this as $a_v^* \in \arg\max_{a \in \mathbb{A}} V(a)$.

The CA *mechanism* defines how the *allocation* is determined, and it specifies a *payment rule*, which defines how bidders' *payments* are determined. In this paper, we describe machine learning-based mechanisms that are agnostic to the specific payment rule used.

3 Machine Learning-Based Mechanisms

The main idea of our machine learning-based auction mechanism is that instead of eliciting the bidders' full value functions, the mechanism aims to only elicit a small number of bids (i.e., value reports for bundles) from each bidder. It then uses a machine learning (ML) algorithm to generalize from those bids to the bidders' full value functions.

3.1 Proxy Agents

To conceptualize the interaction of the bidders with the ML-based mechanism, we assume that each bidder is equipped with a *proxy agent*: each bidder i makes value reports to its proxy agent, and the proxy agent interacts with the mechanism center. We assume that the proxy agents are hosted and operated by the mechanism itself, i.e., this is only a *conceptual tool* and not meant to lead to any separation of agency, or an improvement in terms of privacy or security. See Figure 1 for a high-level depiction of our auction framework.

3.2 Queries and Bidding Language

There are two common preference elicitation queries the proxy could use to interact with the bidder:

- *Value Query:* The proxy queries the bidder with a bundle $s_i \in \{0,1\}^m$ and asks him to report his value $v_i(s_i)$.
- *Demand Query:* The proxy queries the bidder with a price vector specifying a price for each bundle in the bundle space. The bidder replies with the bundle that maximizes his utility at the current prices.

It is not easy to compare the difficulty of answering value or demand queries, because this difficulty depends on the *internal representation* of the bidder's value function (Blumrosen and Nisan 2007). In this work, we decided to let the proxies use value queries to interact with the bidders, because this allows an immediate application of the machine

learning framework. In particular, we obtain a direct parallel between the value-querying process from the CA preference elicitation literature and the generation of a training set via sampling from the ML literature.

3.3 The Proxy's Machine Learning Algorithm

Each proxy agent is equipped with an ML algorithm $\mathcal{A}$.[3] For bidder i, $\mathcal{A}$ will be trained on a vector of data points d_i, e.g., $d_i = (d_{i1}, ..., d_{ik}, ..., d_{i\ell})$. Throughout the paper, we use d_i to denote a generic vector of data points and ℓ to denote a *generic* number of data points. Each data point is a tuple $d_{ik} = (x_{ik}, y_{ik})$, where each x_{ik} is a bundle, and $y_{ik} = v_i(x_{ik})$ is bidder i's value report for bundle x_{ik}.

For our ML-based mechanisms, we distinguish two different phases where the ML algorithm comes into play:

1. an *initial training phase*, where the proxy queries a set of bundles from a given distribution, and
2. an *(optional) iterative phase*, where the proxy may query additional bundles.

In the initial training phase, bidder i's proxy receives exactly q samples $d^0_{i1}...d^0_{ik}...d^0_{iq}$ by sampling q times a bundle x_{ik} from the distribution $\mathcal{X}_i$, without replacement, and querying bidder i for the associated value.[4] We let $\mathcal{X} = \mathcal{X}_1 \otimes \mathcal{X}_2 ... \otimes \mathcal{X}_n$ denote the corresponding joint distribution. We let $d^0_i = (d^0_{i1}, ..., d^0_{iq})$ denote the vector (of length q) of initial training data from bidder i, with $d^0_i \sim \mathcal{D}^0_{\mathcal{X},i}$. We let $\mathcal{D}^0_{\mathcal{X}}$ denote the corresponding joint distribution.

In the (optional) iterative phase, the mechanism may decide to elicit additional information from the bidders. We let d^+_i denote the vector of *additional* data points queried from bidder i (of length q^+_i). Finally, we let $q^\star_i = q + q^+_i$ denote the total number of data points received from bidder i, and $d^\star_i = [d^0_i, d^+_i]^\dagger$ denote the whole vector of data points received from bidder i, with $d^\star_i \sim \mathcal{D}^\star_{\mathcal{X},i}$. We let $\mathcal{D}^\star_{\mathcal{X}}$ denote the corresponding joint distribution. Note that the distribution $\mathcal{D}^\star_{\mathcal{X}}$ is induced by $\mathcal{X}$ *and* by the method which the mechanism uses to query additional bundles in the iterative phase.

Given any vector of training data d_i, $\mathcal{A}$ determines bidder i's *generalized value function* $\tilde{v}_i = \mathcal{A}(d_i)$, which allows it to compute the generalized value $\tilde{v}_i(s_i)$ for each bundle s_i. Based on this, we define the *generalized social welfare* of an allocation a as: $\tilde{V}(a) = \sum_i \tilde{v}_i(a_i)$.

3.4 The Mechanisms

Our ML-based mechanisms have three parameters: the ML algorithm $\mathcal{A}$, the distribution for the initial training phase $\mathcal{X}$, and the number of initial samples q. Consequently, we let $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ denote a mechanism. For now, we leave $\mathcal{A}$ unspecified, but in Section 5.1, we instantiate the ML algorithm with support vector regression (using different kernels). For the rest of the paper, we consider two specific mechanisms:

1. **One-shot Mechanism:** The allocation is determined in one round. In particular, the data available to each bidder's proxy is only collected in one shot.
2. **Iterative Mechanism:** The allocation is determined in multiple rounds, and each bidder's proxy collects data in an iterative fashion, depending on the center's feedback.

Algorithm 1 provides a formal description of the one-shot ML-based mechanism.

Algorithm 1: ML-Based Mechanism: One-Shot Version

Parameters: ML algorithm $\mathcal{A}$, distribution for initial samples $\mathcal{X}$, number of initial samples q

1 For each bidder i, the associated *proxy agent* samples q bundles from $\mathcal{X}_i$ without replacement and queries its bidder for the corresponding values, yielding the initial training data d^0_i.
2 Each proxy agent invokes $\mathcal{A}$ to determine $\tilde{v}_i = \mathcal{A}(d^0_i)$.
3 Each proxy agent submits $\tilde{v}_i$ to the *center*.
4 Allocation $a \in \arg\max_{a \in \mathbb{A}} \sum_i \tilde{v}_i(a_i)$ is selected.
5 Payments are determined by some payment rule.

In Line 2, we see that the proxies only invoke their ML algorithm once - based on the initial training data d^0_i. The proxies only have one interaction with the center (Line 3), which then directly computes the final allocation (Line 4). In particular, the mechanism provides no feedback about the allocation to the bidders, before making the allocation final. The iterative mechanism, provided in Algorithm 2, addresses exactly this shortcoming.

Algorithm 2: ML-Based Mechanism: Iterative Version

Parameters: ML algorithm $\mathcal{A}$, distribution for initial samples $\mathcal{X}$, number of initial samples q

1 For each bidder i, the associated *proxy agent* samples q bundles from $\mathcal{X}_i$ without replacement and queries its bidder for the corresponding values, yielding the initial training data d^0_i.
2 $\theta^1_i \leftarrow d^0_i$ ("current data" = data from training phase).
3 $t \leftarrow 0$; $a^0 \leftarrow$ *null allocation* .
4 **do**
5 $\quad t \leftarrow t + 1$.
6 $\quad$ Each proxy agent invokes ML algorithm $\mathcal{A}$ to determine $\tilde{v}^t_i = \mathcal{A}(\theta^t_i)$.
7 $\quad$ Each proxy agent submits $\tilde{v}^t_i$ to the *center*.
8 $\quad$ Center computes $a^t \in \arg\max_{a \in \mathbb{A}} \sum_i \tilde{v}_i(a_i)$.
9 $\quad$ Center sends a^t_i to each corresponding proxy agent.
10 $\quad$ If a^t_i has not yet been queried from bidder i, bidder i's proxy agent queries $v_i(a^t_i)$ from bidder i and updates $\theta^{t+1}_i = [\theta^t_i, (a^t_i, v_i(a^t_i))]$.
11 **while** $a^{t-1} \neq a^t$.
12 Allocation a^t becomes the final allocation.
13 Payments are determined by some payment rule.

The motivation for using the iterative instead of the one-shot version is to further increase efficiency, by trying to

[3] One could use a different ML algorithm for each proxy. But for simplicity we assume that all proxies use the same $\mathcal{A}$.

[4] The proxies could uses different initial training set sizes. But for simplicity we assume that the same q is used by all proxies.

[†] Note that $[,]$ denotes simple vector concatenation.

correct some of the "mistakes" (overestimates or underestimates) the ML algorithm may make. By querying tentatively allocated bundles we allow for corrections of the generalized valuation in those parts of the value space that are very likely to be relevant for determining the efficient allocation. As we show in Section 6, the iterative version does indeed lead to a significant increase in efficiency.

Remark 1 (Individual Rationality)**.** *ML-based auction formats may not elicit the information necessary to determine payments so that individual rationality (IR) is preserved. To elicit this information it is important that the value for the bundle allocated to each bidder is always queried before determining the payments. However, querying the bundle after the final allocation is determined may lead to serious strategic issues: a bidder could just report a very low value to make sure that he will be charged low payments. The iterative version of our auction overcomes this problem by ensuring that all the necessary information is elicited before the allocation is finalized.*[5] *This ability of the iterative auction to guarantee IR in this way is one of its big advantages over the one-shot auction, and allows the application of common payment rules such as first-price, VCG, core-selecting payment rules, etc., while guaranteeing IR.*[6]

4 Probably Approximate Efficiency

In the mechanisms presented in Section 3.4, the generalized value functions may not be exact, resulting in a loss of economic efficiency when the final allocation is chosen. To capture this, we introduce a new concept called *Probably Approximate Efficiency (PAE)* as a desirable relaxation of the standard efficiency desideratum of mechanism design. For this we need an additional assumption on the valuation profiles v, namely that $v \sim \mathcal{V}$, where $\mathcal{V}$ denotes a joint distribution over possible value functions; we denote by $\mathcal{V}_i$ the corresponding marginal for bidder i. Note that $\mathcal{V}$ captures the randomness in the bidders' preferences over the domain, while $\mathcal{D}^{\star}_{\mathcal{X}}$ captures the randomness over the bundles being sampled from the domain. Our PAE concept captures both of these sources of randomness.

Definition 1 (Probably Approximate Efficiency (PAE))**.** *For a given $\delta \in (0,1)$ and a distribution over valuation profiles $\mathcal{V}$, a mechanism $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ is* probably approximately efficient (PAE) *with bound ε_δ, if it determines an allocation $a^*_{\tilde{v}}$ such that:*

$$\mathrm{P}_{v\sim\mathcal{V}, d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}\left(1 - \frac{V(a^*_{\tilde{v}})}{V(a^*_v)} \geq \varepsilon_\delta\right) \leq \delta. \qquad (1)$$

Here ε_δ is a bound on the efficiency loss valid at least $1-\delta$ fraction of the time. Note that $a^*_{\tilde{v}}$ is the outcome found by optimizing $\tilde{V}(\cdot)$, and thus indirectly depends on $d^\star$.

[5] In practice, we would simply need to save all value reports that the bidders make throughout the auction.

[6] Of course, the fact that the mechanism is iterative rather than one-shot may open up other opportunities for strategic manipulation. However, we leave the analysis of this aspect (incentives and strategic behavior) to future work.

Our new PAE concept enables us to capture the efficiency of ML-based auction mechanisms. However, recall our original design goal, of keeping the interaction between the bidders and the mechanism low - in particular, avoiding the enumeration of the exponentially large bundle space. Thus, in our new formalism, we want mechanisms that a have low $q^\star$. For those mechanisms to be PAE, with a small δ and ε_δ, we need an effective learning algorithm that is able to generalize well from a small number of samples to the whole bundle space. In a first step, we now define a measure for the *generalization error* of an ML algorithm for those valuations likely to arise in the efficient allocation, given $\mathcal{V}$. In a second step, we then relate this generalization error to the efficiency loss of our mechanisms.

In the ML literature, generalization error is often codified as the expected loss over all points outside the training set. Here, we find it more useful to instead bound the error pointwise, using the notion of a *prediction interval*, because we are not interested in *average* error but rather the error at exactly the efficient allocation point. When specialized to our setting, such an interval has the following form:

Definition 2 (Generalized Value Prediction Interval)**.** *For a given $\delta \in (0,1)$, a distribution over valuation profiles $\mathcal{V}$, and a mechanism $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$, the generalized value function $\tilde{v}_i$ for bidder i has an $\eta_{i,\delta}(s)$-prediction interval, if the following holds for all bundles s:*

$$\mathrm{P}_{v_i\sim\mathcal{V}_i,\, d^\star_i\sim\mathcal{D}^\star_{\mathcal{X},i}}\Big(|v_i(s) - \tilde{v}_i(s)| \geq \eta_{i,\delta}(s)\Big) \leq \delta \qquad (2)$$

Informally, this states that with probability $1-\delta$ the prediction at every bundle s will correct up to an error of $\eta_{i,\delta}(s)$. Note that the constant $\eta_{i,\delta}(s)$ will depend on the structure of $\mathcal{V}_i$, the algorithm $\mathcal{A}$ employed to learn it, and $\mathcal{D}^\star_{\mathcal{X},i}$ (and consequently $q^\star$).[7]

Using Definition 2, we can now make a connection between the generalizability of the learning algorithm $\mathcal{A}$ within the setting $\mathcal{V}$ to the eventual efficiency of the mechanism as follows:

Theorem 1. *Suppose the generalized valuation functions in mechanism $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ have a $\eta_{i,\delta}(s)$-prediction interval with respect to distribution over valuation profiles $\mathcal{V}$. Then $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ is PAE with respect to $\mathcal{V}$ with*

$$\varepsilon_\delta = \frac{\eta_\delta(a^*_v) + \eta_\delta(a^*_{\tilde{v}})}{V(a^*_v)} \qquad (3)$$

where $\eta_\delta(a) = \sum_{1\leq i\leq n} \eta_{i,1-\sqrt[n]{1-\delta}}(a_i)$.

Proof. Since $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ has an $\eta_{i,\delta}(s)$-prediction interval with respect to $\mathcal{V}$ we have the following joint bound:

$$\mathrm{P}_{\substack{v\sim\mathcal{V}\\ d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}}\left(\bigwedge_{1\leq i\leq n}(|v_i(a_i) - \tilde{v}_i(a_i)| \leq \eta_{i,\gamma}(a_i))\right) \geq (1-\gamma)^n \forall a \quad (4)$$

[7] Note that this is notionally similar to the *confidence intervals* commonly used in the empirical sciences, however it provides a quantile-based bound at an *out-of-sample* prediction point instead of at an *in-sample* data point.

Because social-welfare is additive we obtain:

$$\mathrm{P}_{v\sim\mathcal{V},d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}\left(\left|V(a)-\tilde{V}(a)\right|\leq\eta_\delta(a)\right)\geq 1-\delta\ \forall a \quad (5)$$

where: $\eta_\delta(a)=\sum_{1\leq i\leq n}\eta_{i,1-\sqrt[n]{1-\delta}}(a_i)$. Substituting in the optimal allocations a^*_v and $a^*_{\tilde{v}}$ then yields:

$$\mathrm{P}_{v\sim\mathcal{V},d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}\left(|V(a^*_v)-\tilde{V}(a^*_v)|\leq\eta_\delta(a^*_v)\right)\geq 1-\delta \quad (6)$$

$$\mathrm{P}_{v\sim\mathcal{V},d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}\left(|V(a^*_{\tilde{v}})-\tilde{V}(a^*_{\tilde{v}})|\leq\eta_\delta(a^*_{\tilde{v}})\right)\geq 1-\delta \quad (7)$$

Because the points are optimal, the objective can only be off by the combined amount:

$$\mathrm{P}_{v\sim\mathcal{V},d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}\left(V(a^*_v)-V(a^*_{\tilde{v}})\leq\eta_\delta(a^*_v)+\eta_\delta(a^*_{\tilde{v}})\right)\geq 1-\delta \quad (8)$$

Finally, dividing by the social welfare and performing some algebra yields:

$$\mathrm{P}_{v\sim\mathcal{V},d^\star\sim\mathcal{D}^\star_{\mathcal{X}}}\left(1-\frac{V(a^*_{\tilde{v}})}{V(a^*_v)}\geq\frac{\eta_\delta(a^*_v)+\eta_\delta(a^*_{\tilde{v}})}{V(a^*_v)}\right)\leq\delta \quad (9)$$

This provides the result. □

5 SVR-Based Auction Mechanisms

5.1 Support Vector Regression

To obtain a generalized value function $\tilde{v}_i$ from training data d_i (i.e., a vector of bundle-value pairs), SVR projects any input bundle s_i into a high dimensional *feature space* via a mapping function $\varphi(s_i)$, and then the prediction $\tilde{v}_i(s_i)$ is a weighted linear combination of the training data in the feature space (see Smola and Vapnik (1997) for details on SVRs). To find the weights, the method simultaneously maximizes the *generalizability* of the learned function through a regularization term that minimizes the sum-squared weights, and minimizes the prediction error on the training data. Formally we have:

$$\min_{w,b,e}\frac{1}{2}w^Tw+c\sum_{k=1}^{\ell}\mathcal{L}(e_k) \quad (10)$$
$$\text{s.t. } y_{ik}=w^T\varphi(x_{ik})+b+e_k\ \forall\ 1\leq k\leq\ell$$

where w are the weight variables, b the offset term, the e_k are the prediction errors on the training data, and $\mathcal{L}(e_k)$ is the loss due to the prediction error of sample k. Traditionally SVR uses the ζ-insensitive loss with $\mathcal{L}(e_k)=\max(0,|e_k|-\zeta)$. Specializing to our setting, SVR then predicts $\tilde{v}_i$ at a bundle s_i as:

$$\tilde{v}_i(s_i)=\sum_{k=1}^{\ell}\alpha_{ik}\kappa(s_i,x_{ik})+b_i \quad (11)$$

where α_i and b_i are the solution to the *dual* of Problem (10), and $\kappa(s_a,s_b)=\langle\varphi(s_a),\varphi(s_b)\rangle$ is a *kernel*, which via the *kernel trick* is computable in closed form for suitable choices of κ and φ (Vapnik and Vapnik 1998). As we can see from (10), only the bundles x_{ik} with associated coefficient $\alpha_{ik}\neq 0$ contribute to the determination of the predicted value for a new input. These bundles are called support vectors.

5.2 Expressiveness

Next, we consider how the choice of kernel influences the types of value functions that can be perfectly captured by our mechanism.

Definition 3 (Expressive ML-Based Mechanism)**.** *An ML-based mechanism $\mathcal{M}(\mathcal{A},\mathcal{X},q)$ is* expressive *for a distribution over valuation profiles $\mathcal{V}$ if, for every bidder i, every v_i in the support of $\mathcal{V}_i$, and every vector of data points $d^\star_i\sim\mathcal{D}^\star_{\mathcal{X},i}$, the mechanism's ML algorithm $\mathcal{A}$ determines a generalized value function $\tilde{v}_i=\mathcal{A}(d_i)$, such that:*

$$\tilde{v}_i(x_{ik})=v_i(x_{ik})\ \forall\ 1\leq k\leq q^\star_i \quad (12)$$

where $d^\star_i=(d^\star_{i1},...d^\star_{ik},...,d^\star_{iq^\star_i})$ and $d^\star_{ik}=(x_{ik},v_i(x_{ik}))$.

We will say that a kernel is expressive, if the kernel induces an *expressive SVR-based* mechanism. Formally:

Definition 4 (Expressive Kernel)**.** *A kernel is* expressive *for a distribution over valuation profiles $\mathcal{V}$ if we can use the kernel to construct an SVR Ξ such that, for all $\mathcal{X}$ and q, the mechanism $\mathcal{M}(\Xi,\mathcal{X},q)$ is* expressive *for $\mathcal{V}$.*

Informally, a kernel is expressive if the induced SVR is able to exactly capture the bidder's reported values (i.e., with zero in-sample error). Using this concept, we now consider several kernels for use in our mechanism. We start by generalizing the standard polynomial kernel to include weights:

Definition 5 (Generalized Polynomial Kernel)**.** $\kappa_p(s_a,s_b)=\sum_{r=0}^{\bar{r}}\lambda_r\langle s_a,s_b\rangle^r$ *is a* generalized polynomial kernel of degree $\bar{r}$ *where $\bar{r}\in\mathbb{N}$, $\lambda_r\geq 0\ \forall\ 0\leq r<\bar{r}$, and $\lambda_{\bar{r}}>0$.*

For such kernels, we have:

Proposition 1 (Expressiveness of Generalized Polynomial Kernels)**.** *The generalized polynomial kernel of degree $\bar{r}$ is expressive for all the distributions $\mathcal{V}$ where each v_i is drawn from the class of $\bar{r}$-wise dependent valuations (Conitzer, Sandholm, and Santi 2005).*

Proof. To guarantee expressivity it is sufficient to find a feature mapping φ_p such that $\kappa_p(s_a,s_b)=\varphi_p(s_a)^T\varphi_p(s_b)\ \ \forall\ \ s_a,s_b$ and check that constraints in Equation 10 $w^T\varphi_p(x_{ik})+b=y_{ik}\ \forall\ 1\leq k\leq\ell$ hold exactly ($e_k=0\ \forall\ k$) for at least one solution (w,b). A polynomial kernel of degree $\bar{r}$ manifests a feature mapping φ_p with a feature z for every subset of items $\psi:|\psi|\leq\bar{r}$, and thus for any given bundle s_i, $\varphi_p(s_i)_z>0$ if s_i contains all the items in ψ, and 0 otherwise (Sec. 3.4 Shawe-Taylor and Cristianini 2004). Since the y_{ik} are reported from an $\bar{r}$-wise dependent valuation there will be a solution to the system as follows: $b=0$ and each w_z is set such that $w_z\varphi(s)_z$ contributes the value due to the interdependency among the items in ψ. □

Next, we derive the following general result:

Proposition 2 (Expressiveness of Strictly Positive-Definite Kernels)**.** *Strictly positive definite kernels are expressive for any distribution of valuation profiles $\mathcal{V}$.*

Proof. Considering Equation (11), we can rewrite Equation (12) as a linear system: $K\alpha_i + b_i\vec{1}_\ell = y_i$ where $K_{ab} = \kappa(x_{ia}, x_{ib})$ is the kernel matrix on the training data, $\alpha_i = (\alpha_{i1}, \ldots, \alpha_{i\ell})$ and $\vec{1}_\ell$ is a vector if ℓ ones. The matrix K related to any strictly positive definite kernel has full rank (Hofmann, Schölkopf, and Smola 2008). Thus, the linear system always has a solution. □

We next consider two important kernels that belong to this class. We start with the exponential kernel which is the limit case of the generalized polynomial kernel where $\lambda_r = {\lambda_e^r}/{r!}$ for a new constant $\lambda_e > 0$:

Corollary 1 (Expressiveness of the Exponential Kernel)**.** *The exponential kernel $\kappa_e(s_a, s_b) = \exp(\lambda_e \langle s_a, s_b \rangle)$ is expressive for any distribution over valuation profiles $\mathcal{V}$.*

Corollary 2 (Expressiveness of the Gaussian Kernel)**.** *The Gaussian kernel $\kappa_g(s_a, s_b) = \exp({-||s_a - s_b||}/{\lambda_g})$ is expressive for any distribution over valuation profiles $\mathcal{V}$.*

5.3 Consistency

We can now bolster the efficiency result from Section 4 with a desirable limit condition.

Definition 6 (Probably Approximately Consistent (PACO))**.** *An ML-based mechanism $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ is* probably approximately consistent (PACO) *if, when enough data is drawn such that $\min_i q_i^\star = 2^m$, then, for every $\delta \in (0, 1)$, the mechanism is PAE with $\varepsilon_\delta = 0$.*

In words, PACO requires the simple consistency property that the mechanism has no learning error and thus is fully efficient if the full bundle space has been queried. We can instantiate this definition to expressive kernels as follows:

Theorem 2. *An ML-based mechanism $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$ that uses an SVR with an* expressive *kernel as its learning algorithm $\mathcal{A}$, a sufficiently large regularization constant c, and sets the ζ-insensitive loss threshold to 0 is PACO.*

Proof. By Definition 4, when the SVR uses an expressive kernel, and fully optimizes for accuracy on the training data by setting a sufficiently large c and $\zeta = 0$, it will then predict exactly on all the training points it is given. Since when $q_i^\star = 2^m$ the data becomes complete, it will exactly match each point, driving $\eta_\delta(s_i) \to 0$ for all s_i. The result then follows from Theorem 1. □

5.4 Winner Determination

The ability to concisely obtain generalized value functions in the proxies is only useful to the extent that we can then solve the winner determination problem with respect to them. Fortunately, it is possible to encode a winner determination problem based on the functions learned under our chosen kernels, $\tilde{v}(s_i)$, as an integer program (IP). That we can do this is perhaps surprising, as it requires us to compactly encode our *non-linear* social welfare objective in a *linear* program, a feat we manage by exploiting the structure of the kernel functions we employ. The effect is that we are able to solve a problem with a non-linear objective, while still using standard IP solvers (such as CPLEX). This provides for fast solution times, because we place only a modest overhead above the fundamental combinatorial decision that must be made in any CA winner determination algorithm. Our IP formulation begins as follows:

$$\begin{aligned}
\underset{A_{ij}, G_{ik\tau}}{\arg\max} \quad & \sum_{i=1}^{n} \sum_{k=1}^{\ell} \alpha_{ik} \sum_{\tau=0}^{\bar{\tau}} \breve{\kappa}(\tau) G_{ik\tau} && \text{Social Welfare} \\
\text{s.t.} \quad & \sum_{\tau=1}^{\bar{\tau}} G_{ik\tau} = 1 \; \forall i, k && \text{Eval. } \breve{\kappa} \text{ at one pt.} \\
& \sum_{i=1}^{n} A_{ij} \leq 1 \; \forall j && \text{Supply} \\
& A_{ij}, G_{ik\tau} \in \{0, 1\}
\end{aligned}$$

Here, the main decision variable $A_{ij} = 1$ iff bidder i obtains good j in the optimal allocation. The social welfare objective is then formed as a sum over each bidder's support vectors indexed by k. For each, the value of their α_{ik}-weighted kernel value is computed. For the $\{0, 1\}$ input space present in our bundle-based problem, the kernels we have presented consist of the composition of a non-linear function $\breve{\kappa} : \mathbb{R} \mapsto \mathbb{R}$, with a linear one $\bar{\kappa} : \mathbb{W} \mapsto \mathbb{R}$ so that $\kappa(s_a, s_b) = \breve{\kappa}(\bar{\kappa}(s_a, s_b))$. We take advantage of this to linearize the kernel in the objective: we calculate the constant appropriate to the kernel $\breve{\kappa}(\tau)$, and then gate its inclusion in a sum over all $0 \leq \tau \leq \bar{\tau}$ for that support vector with a binary auxiliary variable $G_{ik\tau}$ such that $G_{ik\tau} = 1$ iff $\tau = \bar{\kappa}(s_a, s_b)$. We then enforce that there is only one active $G_{ik\tau}$ per support vector, and the standard supply constraint in the allocation variables. All that remains is to include an explicit constrant that enforces our stipulation that $G_{ik\tau} = 1$ iff $\tau = \bar{\kappa}(x_{ik}, A_i)$. This constraint is specific to the kernel choice.

For polynomial and exponential kernels we set $\bar{\tau} \leftarrow |x_{ik}|$, and use $\bar{\kappa}(x_{ik}, A_i) \leftarrow \langle x_{ik}, A_i \rangle$, which is formulated as:

$$\sum_{j \in x_{ik}} A_{ij} = \sum_{\tau=0}^{\bar{\tau}} (\tau + 1) G_{ik\tau} - 1 \; \forall i, k \tag{13}$$

Further, we set $\breve{\kappa}(\vartheta) \leftarrow \sum_{r=0}^{\bar{r}} \lambda_r \vartheta^r$ for the polynomial kernel and $\breve{\kappa}(\vartheta) \leftarrow \exp(\lambda_e \vartheta)$ for the exponential kernel.

For the Gaussian kernel, $\bar{\tau} \leftarrow m$ and $\bar{\kappa}(x_{ik}, A_i) \leftarrow \sum_{1 \leq j \leq m} x_{ikj} \oplus A_{ij}$, which is formulated as:

$$\sum_{j \in x_{ik}} (1 - A_{ij}) + \sum_{j \notin x_{ik}} A_{ij} = \sum_{\tau=0}^{\bar{\tau}} (\tau + 1) G_{ik\tau} - 1 \; \forall i, k \tag{14}$$

Further, we set $\breve{\kappa}(\vartheta) \leftarrow \exp({-\vartheta}/{\lambda_g})$

With the inclusion of an appropriate choice of Equation (13) or Equation (14), our formulation can then encode all of the kernels described above. We note that, the number of support vectors greatly influences the complexity of the winner determination problem in our formulation. A way to control for this affect is to use the ζ-insensitive loss with a positive ζ to decrease the number of support vectors.

6 Experimental Analysis

6.1 Experimental Setup

To evaluate the effectiveness of our proposed mechanisms, we have conducted a set of experiments to empirically identify the smallest PAE bound ε_δ for different distributions over value functions $\mathcal{V}$, and for several ML-based mechanisms under different choices of inputs $\mathcal{A}$, $\mathcal{D}$, and q.

Because spectrum auctions are a very important domain for CAs, we have chosen to evaluate our mechanisms in this context. We use two standard value models from the literature, both of which capture a stylized version of the geographic structure in large spectrum auctions. Both include multiple bidder types, with some interested in larger packages, and others interested in smaller packages. The Global Synergy Value Model (*GSVM*) (Goeree and Holt 2008) models the goods (spectrum licenses) as being arranged in two circles. Depending on his type, a bidder may be interested in licenses from different circles, and has a value that depends on the total number of licenses of interest. We let $\mathcal{V}_{\text{GSVM}}$ denote the distribution over valuation profiles corresponding to GSVM. By contrast, the more complex Local Synergy Value Model (*LSVM*) (Scheffel, Ziegler, and Bichler 2012) places the goods on a two-dimensional grid. Here, a bidder's value depends on a sigmoid function of the number of *contiguous* licenses. We let $\mathcal{V}_{\text{LSVM}}$ denote the distribution over valuation profiles corresponding to LSVM.

Within these domains, we have evaluated several mechanisms. First, to provide a baseline to quantify the benefits of using ML, we also include in our experiments a simple mechanism (*No ML*), where the proxies report the bundle-value pairs obtained from the bidders as XOR bids to the center, without first using ML to generalize from this data.

Next, we have considered both *One-Shot* mechanisms (Algorithm 1) and *Iterative* mechanisms (Algorithm 2). For those ML-based mechanisms $\mathcal{M}(\mathcal{A}, \mathcal{X}, q)$, in all of our experiments, we used an SVR for $\mathcal{A}$; this in turn is parameterized by the choice of kernel, regularization constant c, and ζ-insensitive loss threshold. In our experiments we set $c = 100$ and $\zeta = 0.1$ such that the minimization of the interpolation loss is prioritized in the objective of (10).

For both $\mathcal{V}_{\text{GSVM}}$ and $\mathcal{V}_{\text{LSVM}}$ we will consider the exponential and Gaussian kernels, since (as we have shown) they are always expressive. For $\mathcal{V}_{\text{GSVM}}$ we will additionally consider the generalized polynomial kernel of degree 2, as it can be shown that this kernel is expressive for $\mathcal{V}_{\text{GSVM}}$. The kernels have hyperparameters λ_e, λ_r and λ_g respectively. We tuned these parameters from a hold-out set of 100 valuation profiles from the respective setting under consideration.

We consider two different *initial training distributions* $\mathcal{X}$. We first study *Full-Random*, where each $\mathcal{X}_i$ is a uniform distribution over the full bundle space, which implies that a bidder may (likely) be asked to report a value for a bundle in which he has no interest. This models a situation where neither the agent nor the center has any useful prior over the bidder's value structure. Secondly, we study *Pos-Random*, where each $\mathcal{X}_i$ is a uniform distribution over bidder i's *bundles of interest*.[8] This models the case where there is a simple prior that partitions the bundle space into bundles with and without interest. For example if we have an "east cost bidder" we know all the "west coast" bundles are not of interest. Such a partition is often available in CA settings. Finally, we also vary the number of initial draws from the distribution $\mathcal{X}$, $q \in \{10, 20, 40\}$.

Initial Sample Distribution	Mechanism	Kernel	Efficiency †		
			$q=10$	$q=20$	$q=40$
Full-Random	No ML	-	34%	39%	43%
	One-Shot	Poly-2	59%	80%	**92%**
		Exp	57%	70%	86%
		Gaus	52%	70%	83%
	Iterative	Poly-2	**74%**	**86%**	**92%**
		Exp	66%	79%	88%
		Gaus	71%	79%	86%
Pos-Random	No ML	-	75%	82%	88%
	One-Shot	Poly-2	89%	96%	**100%**
		Exp	89%	95%	99%
		Gaus	80%	84%	86%
	Iterative	Poly-2	91%	**97%**	**100%**
		Exp	**94%**	**97%**	**100%**
		Gaus	86%	87%	87%

Table 1: Efficiency results for the Global Synergy Value Model under a variety of sample distributions, mechanisms, kernels and initial queries q, over 100 auctions. The reported values are the 10th-quantile of efficiency, i.e. 90% of the evaluated auctions had an efficiency at least this high.

6.2 Efficiency Results

To find the smallest PAE bound ε_δ empirically, we ran 100 different auction instances in each setting, and measured the efficiency obtained in each. The empirical estimate of ε_δ is then simply the δ-quantile of the distribution of these efficiencies. In Tables 1 and 2 we present the results for $\delta = 0.1$; we can therefore interpret the numbers provided as saying that 90% of the auction instances were at least this efficient.

Table 1 provides the smallest PAE bound $\varepsilon_{0.1}$ for the GSVM. We observe that our one-shot ML-based mechanism significantly out-performs the No-ML mechanism that submits XOR bids of the same bundles. We also observe that our iterative mechanism always outperforms the one-shot version. Further, we find that the exponential and the generalized polynomial kernel of degree 2 perform extremely well, and in particular better than the Gaussian kernel. We observe broadly similar *patterns* in the Local Synergy Value Model presented in Table 1. However, on average, all mechanisms show a lower efficiency under LSVM than under GSVM.[9]

[8] Note that under GSVM and LSVM, the bundles of interest correspond to the set of bundles where the bidder has positive value for each individual item in each bundle.

[†] Since in the Iterative auction designs the final number of queried bundles will be larger than the initial number q, we note that the average $\Delta q = q^\star - q^0$ of the iterative auctions is 3.0 for the GSVM and 4.6 for the LSVM.

[9] One reason is the complex object topology of the LSVM (i.e., the two-dimensional grid), which is not explicitly captured by any of our kernels, and may hinder generalizability.

Initial Sample Distribution	Mechanism	Kernel	Efficiency †		
			$q=10$	$q=20$	$q=40$
Full-Random	No ML	-	34%	38%	43%
	One-Shot	Exp	56%	64%	70%
		Gaus	32%	43%	51%
	Iterative	Exp	**70%**	**73%**	**77%**
		Gaus	50%	59%	64%
Pos-Random	No ML	-	60%	68%	75%
	One-Shot	Exp	79%	83%	82%
		Gaus	54%	65%	66%
	Iterative	Exp	**88%**	**92%**	**92%**
		Gaus	63%	65%	70%

Table 2: Efficiency results for the Local Synergy Value Model (LSVM) under a variety of sample distributions, mechanisms, kernels and initial queries q, over 100 auction trials. The reported values are the 10th-quantile of efficiency, i.e. 90% of the auctions had an efficiency at least this high.

6.3 Sensitivity to Hyperparameters

To test the sensitivity of the results to the tuning of the hyperparameters, we ran a separate experiment where we tuned the hyperparameters using data from the wrong model. Table 3 shows results for $\mathcal{V}_{\text{GSVM}}$ with hyperparameters optimized for $\mathcal{V}_{\text{LSVM}}$ and vice-versa. We observe that in all cases the mechanism is still able to achieve high efficiency, even when the hyperparameters have been set based on wrong training data. While all of the kernels are remarkably robust to this effect, we see that the Gaussian kernel is especially so. The *average efficiency loss* was 1.8 for all $\mathcal{V}_{\text{LSVM}}$-based settings, and 2.2 for all $\mathcal{V}_{\text{GSVM}}$-based settings.

6.4 Trade-off Between Efficiency and Runtime

As mentioned in Section 5.4, increasing the ζ-insensitivity threshold ζ above zero decreases the number of support vectors, and therefore may stand to improve the speed of winner determination, at the possible loss of some efficiency. To test this hypothesis, we ran an experiment where we varied

Domain & Initial Sample Distribution	Mechanism	Kernel	Efficiency Loss †		
			$q=10$	$q=20$	$q=40$
GSVM Full-Random	Iterative	Exp	-6%	-13%	-0%
		Gaus	-0%	-0%	-0%
GSVM Pos-Random	Iterative	Exp	-0%	-0%	-0%
		Gaus	-0%	-0%	-0%
LSVM Full-Random	Iterative	Exp	-9%	-5%	-5%
		Gaus	-0%	-0%	-0%
LSVM Pos-Random	Iterative	Exp	-4%	-5%	-2%
		Gaus	-1%	-1%	-0%

Table 3: Results on the sensitivity to hyperparameters for LSVM and GSVM when the hyperparameters have been tuned for the other domain. The values are the difference in the 10th-quantile of efficiency between the matching experiments with correctly and incorrectly tuned hyperparameters.

Initial Sample Distribution	Kernel	Insensitivity Threshold			
		$\zeta = 0.1$		$\zeta = 10$	
		Eff.	Runtime (s)	Eff.	Runtime (s)
Full-Random	Exp	77%	484.1	74%	53.5
Full-Random	Gaus	64%	369.0	69%	106.4
Pos-Random	Exp	92%	6.1	91%	2.8
Pos-Random	Gaus	70%	334.0	68%	29.7

Table 4: Efficiency and run-time performance when varying ζ-insensitive loss parameter. Shown are the results for the iterative version of the ML-based auction in LSVM with $q = 40$. Results are averaged over 100 trials.

$\zeta \in \{.1, 10\}$ and recorded both the average runtime needed to clear the market and the average efficiency in each setting. These results are shown in Table 4. We observe that increasing ζ does indeed have the effect of reducing the number of support vectors (not shown in Table 4), which leads to a corresponding reduction in WD complexity and increase in computational speed. The observed efficiency decrease is small for a very significant increase in speed.[10] While our IP-based winner determination method can clear reasonable instances quickly, if users of our SVR-based mechanisms require even more clearing-speed, these results indicate that an increase in ζ can produce a dramatic reduction in computational cost for only a modest decrease in efficiency, which in some settings may be an attractive trade-off.

7 Conclusion

In this paper, we have introduced a new auction design paradigm based on machine learning (ML). Our main goal was to reduce the communication burden for the bidders, only eliciting a small number of value reports, and then generalizing to the whole value space using an ML algorithm.

We have introduced a new efficiency measure called *Probably Approximate Efficiency (PAE)*, as a useful relaxation of the standard efficiency notion employed in mechanism design. The new PAE concept enables us to formally capture the efficiency of ML-based mechanisms in a probabilistic way. To instantiate our new auction design paradigm, we have used SVRs as our ML algorithm, with three different kernels: polynomial, exponential, and Gaussian. We have shown that the exponential and Gaussian kernels are fully expressive and that they are consistent in the limit (when querying the full bundle space). For all of the kernels we have investigated, we have shown that the winner determination problem can be succinctly encoded as an IP, which enables us to find the efficient allocation using standard optimization software. Our experimental results comparing our ML-based auctions to standard (non-ML) auctions are encouraging. We have shown that already with a small number of sampled bundles, we can achieve high efficiency, and we

[10] For the Full-Random/Gaus setting, we actually observe an efficiency *increase*. Note that this is consistent with the SVR set-up, given that $\zeta = 0.1$ was not chosen to optimize efficiency. However, on average, the highest efficiency results are expected to be obtained with small ζ's.

are always able to beat the non-ML benchmark.

References

Abernethy, J.; Lahaie, S.; and Telgarsky, M. 2016. Rate of Price Discovery in Iterative Combinatorial Auctions. In *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*.

Afriat, S. N. 1967. The Construction of Utility Functions from Expenditure Data. *International economic review* 8(1):67–77.

Blumrosen, L., and Nisan, N. 2007. Combinatorial auctions. In Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V., eds., *Algorithmic Game Theory*. Cambridge University Press. 276–298.

Blumrosen, L., and Nisan, N. 2009. On the Computational Power of Demand Queries. *SIAM Journal on Computing* 39(4):1372–1391.

Conitzer, V.; Sandholm, T.; and Santi, P. 2005. Combinatorial Auctions with k-Wise Dependent Valuations. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI)*, volume 5, 248–254.

Cramton, P. 2013. Spectrum Auction Design. *Review of Industrial Organization* 42(2):161–190.

Goeree, J. K., and Holt, C. A. 2008. Hierarchical Package Bidding: A Paper & Pencil Combinatorial Auction. *Games and Economic Behavior*.

Goetzendorf, A.; Bichler, M.; Shabalin, P.; and Day, R. W. 2015. Compact Bid Languages and Core Pricing in Large Multi-item Auctions. *Management Science* 61(7):1684–1703.

Hofmann, T.; Schölkopf, B.; and Smola, A. J. 2008. Kernel Methods in Machine Learning. *The annals of statistics* 1171–1220.

Kearns, M. J., and Vazirani, U. V. 1994. *An Introduction to Computational Learning Theory*. MIT press.

Lahaie, S., and Parkes, D. C. 2004. Applying Learning Algorithms to Preference Elicitation. In *Proceedings of the 5th ACM conference on Electronic commerce*, 180–188. ACM.

Lahaie, S. 2009. A Kernel Method for Market Clearing. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 208–213.

Lahaie, S. 2010a. Kernel Methods for Revealed Preference Analysis. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, 439–444.

Lahaie, S. 2010b. Stability and Incentive Compatibility in a Kernel-Based Combinatorial Auction. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*.

Lahaie, S. 2011. A Kernel-Based Iterative Combinatorial Auction. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*.

Sandholm, T. 2013. Very-Large-Scale Generalized Combinatorial Multi-Attribute Auctions: Lessons from Conducting $60 Billion of Sourcing. In Vulkan, N.; Roth, A. E.; and Neeman, Z., eds., *The Handbook of Market Design*. Oxford University Press. chapter 1.

Scheffel, T.; Ziegler, G.; and Bichler, M. 2012. On the Impact of Cognitive Limits in Combinatorial Auctions: An Experimental Study in the Context of Spectrum Auction Design. *Experimental Economics* 15:667–692.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge university press.

Smola, A., and Vapnik, V. 1997. Support Vector Regression Machines. *Advances in neural information processing systems* 9:155–161.

Vapnik, V. N., and Vapnik, V. 1998. *Statistical Learning Theory*, volume 1. Wiley New York.