Machine Learning-powered Iterative Combinatorial Auctions*

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

Benjamin Lubin
Boston University
blubin@bu.edu

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

October 18, 2019

Abstract

In this paper, we present a machine learning-powered iterative combinatorial auction (CA). The main goal of integrating machine learning (ML) into the auction is to improve preference elicitation, which is a major challenge in large CAs. In contrast to prior work, our auction design uses *value queries* instead of prices to drive the auction. The ML algorithm is used to help the auction decide which value queries to ask in every iteration. While using ML inside an auction introduces new challenges, we demonstrate how we obtain a design that is individually rational, has good incentives, and is computationally practical. We benchmark our new auction against the well-known combinatorial clock auction (CCA). Our results indicate that, epecially in large domains, our ML-powered auction can achieve higher allocative efficiency than the CCA, even with only a small number of value queries.

Keywords: Combinatorial Auctions, Machine Learning, Preference Elicitation, CCA

1. Introduction

Combinatorial auctions (CAs) are used to allocate multiple items among bidders who may view these items as complements or substitutes. Specifically, they allow bidders to submit bids on *bundles* of items to express their complex preferences. CAs have found widespread real-world applications, including for the sale of spectrum licenses (Cramton, 2013) and the allocation of TV-ad slots (Goetzendorf et al., 2015).

One of the main challenges when conducting CAs in practice is that the bundle space grows exponentially in the number of items, which typically makes it impossible for bidders to report their full value function. Practical CA designs allow bidders to report only a limited number of bids which are treated as "all-or-nothing" bundle bids by the auctioneer (see, e.g., Ausubel and Baranov (2017)). However, this number can be very small compared to the total number of bundles. For example, the 2014 Canadian spectrum auction (Industry Canada, 2013) had 106 clock rounds (where each bidder essentially submits one bid per round), followed by a second phase in which bidders were limited to submit bids on at most 500 of the 2^{98} possible bundles (and multiple bidders reached this limit of 500 bids). Thus, preference elicitation remains a primary problem, especially in large CAs.

*This paper is a significantly extended version of Brero, Lubin and Seuken (2018) which was published in the conference proceedings of IJCAI-ECAI'18. Some of the ideas presented in this paper have also previously been described in Brero, Lubin and Seuken (2017), which appeared in the conference proceedings of AAAI'17. Part of this research was supported by the SNSF (Swiss National Science Foundation) under grant #156836.

1.1. Preference Elicitation in CAs

Researchers have addressed this preference elicitation challenge by designing *bidding languages* that are succinct yet expressive for specific (restricted) classes of valuation functions (see Nisan (2006) for a survey). For some applications (e.g., in procurement auctions), practitioners have also developed domain-specific bidding languages exploiting specific knowledge about the structure of the bidders' value functions (Sandholm, 2013; Goetzendorf et al., 2015). However, in many domains, like spectrum auctions, bidders' value functions exhibit such a rich and complex structure that they do not fall into any of the succinctly expressible valuation classes supported by the domain-specific languages proposed in the literature. Furthermore, any particular choice of a domain-specific bidding language may favor one bidder over another, which is problematic for a government-run auction which should not bias against certain participants. But, unfortunately, if one must support general valuations and guarantee full (or even approximate) efficiency, then the auction requires an exponential amount of communication in the worst case (Nisan and Segal, 2006).

To get around this difficulty in practice, the preference elicitation challenge is often addressed by using *iterative combinatorial auctions*, where the auctioneer heuristically elicits information from the bidders over multiple rounds, imposing some kind of limit on the amount of information exchange. Common formats include *ascending-price auctions* as well as *clock auctions*, where prices are used to coordinate the bidding process. In recent years, the *combinatorial clock auction (CCA)* has gained momentum (Ausubel et al., 2006). Between 2012 and 2014 alone, ten countries have used the CCA, raising approximately $20 billion in revenues, with the 2014 Canadian 700 MHZ auction being the largest auction, raising more than $5 billion (Ausubel and Baranov, 2017). The CCA consists of two main phases: in the initial clock phase, the auctioneer quotes (linear) item prices in each round, and bidders are asked to respond to a *demand query*, stating their profit-maximizing bundle at the quoted prices. In the second phase (the supplementary round), bidders can submit a finite number of bundle bids (typically up to 500). This design is meant to combine good price discovery in the clock phase with good expressiveness in the supplementary round.

In spite of the practical success of the CCA, a recent line of papers has revealed some of its shortcomings. Regarding bidding complexity, Knapek and Wambach (2012) discuss why bidding optimally in the CCA may be challenging. Janssen, Karamychev and Kasberger (2017) and Janssen and Kasberger (2015) discuss how bidders can engage in strategic behavior if they want to increase rivals' costs or if they are facing budget constraints.

In addition to these issues with bidding complexity and incentives, another line of research has studied the efficiency of the CCA. Some experimental studies have shown that bidders may not be able to accurately respond to a demand query (which may require full exploration of the exponentially-sized value space). In particular, Scheffel, Ziegler and Bichler (2012) and Bichler, Shabalin and Wolf (2013) have shown that bidders tend to focus on a limited search space consisting of some bundles of items selected prior to the auction, and that this can cause significant efficiency losses (between 4% to 11% in their experiments).

1.2. Overview of our Contributions

To address these shortcomings of the CCA, we propose a new auction design, specifically a *machine learning-powered iterative combinatorial auction.* In contrast to the CCA, our auction does not use demand queries (i.e., prices) but instead asks bidders for *value reports* on individual bundles. While in the clock phase of the CCA, clock prices do the job of coordinating bidders towards finding an (approximately) efficient allocation, in our approach, we use an elicitation process guided by machine learning (ML) to serve this role.

In Section 3, we present the design of our new auction which consist of two separate algorithmic contributions. Our first contribution is an *ML-powered QueryModule* (Section 3.1) which, given a set of bundle-value pairs reported by each bidder, determines a new query for each bidder to be answered in the next round of

the auction. The QueryModule consists of two key steps. First, given a set of bundle-value reports for each individual bidder, we use an ML algorithm for each bidder to compute an *inferred value function* for this bidder, which provides us with a prediction of this bidder's value for any bundle in the bundle space. We then compute an efficient allocation based on these inferred valuations, and we use the information from this step to determine the next query for each bidder.

Our second algorithmic contribution is the design of the complete ML-based ICA which we call *Pseudo VCG Machine Learning-based (PVML)* mechanism (Section 3.2). PVML uses the QueryModule as a sub-routine in every round of the auction and ultimately determines an allocation and payments. PVML is built around two main ideas. We charge "VCG-style payments" and we allow bidders to "push" information to the auction which they deem useful for efficiency. Together, these two design features lead to good incentives in practice.

In Section 4 we provide a detailed theoretical analysis of the properties of PVML. We begin by showing how the performance of the ML algorithm, measured in terms of the learning error, relates to the performance of PVML (Section 4.1). Specifically, given a bound on the learning error of the ML algorithm, we provide a bound on the efficiency loss incurred by PVML. Furthermore, we provide a *price-based interpretation* of the elicitation performed by PVML. Even though the bidders only communicate with the mechanism via value reports, we can *impute prices* to gain insight into how bundles are *implicitly priced* as the auction proceeds. We then show that PVML obtains an approximate competitive equilibrium, where the approximation depends on the learning error of the ML algorithm.

We then move on to analyze standard mechanism design properties of PVML. In terms of incentives, we first show that PVML is not ex-post Nash incentive compatible. Similar to other practical CAs like the CCA, this is due to the rich dynamic strategy space of the bidders and the fact that we cannot guarantee finding the efficient allocation. However, we show that PVML is *social welfare aligned* which means that any beneficial manipulation a bidder finds must increase the social welfare by the same amount. Together with the fact that bidders can "push" self-selected bundle-value pairs to the auction in an initial phase, we argue that this provides PVML with "good incentives in practice." Concretely, if bidders have certain beliefs about which bundles they should receive as part of a (highly) efficient allocation, then they can simply push those bundles in the initial phase. Afterwards, they should simply answer all value queries truthfully, to enable the mechanism to maximize social welfare (which it is designed to do).

In Section 4.3, we show that PVML satisfies *individual rationality*, i.e., no bidder obtains negative utility by participating in the auction. Finally, in Section 4.4., we study whether PVML can run at a deficit or not. We show that, in principle, it may run at a deficit - although the probability of a deficit occuring depends on how much information the auctioneer elicits from each bidder (and thus, how well she can compute the Pseudo VCG payments). We then explain that, in practice, an auctioneer who wants to guarantee the *no-deficit* property can simply lower-bound each bidder's payment by zero (albeit, at the consequence of slightly relaxing the social welfare alginment property).

While for the description of PVML in Section 3 and the theoretical analysis in Section 4, we can leave the ML algorithm unspecified, we finally instantiate it in Section 5. In principle, any ML algorithm could be used within PVML. However, a key requirement is that one can relatively quickly solve the auction's *winner determination problem (WDP)* based on bidders' inferred valuations, to compute a new allocation – and this computation is done many hundreds of times (inside the QueryModule) throughout one run of PVML. Thus, in our experiments, we limit the execution time for one such WDP computation to 1 minute. In Section 5, we first explain how standard *linear regression* can be used to obtain the bidder's inferred valuations, and how formulating the linear regression-based WDP is straightforward. Then we generalize this to non-linear machine learning models that can also capture substitutes and complements valuations – in particular, we use *kernelized support vector regression (SVR)*. In this paper, we discuss SVRs with four popular kernels: Linear (which corresponds to linear regression), Quadratic, Exponential, and Gaussian, and we provide WDP formulations for all four. Using a different kernel in the SVR essentially leads to a different ML algorithm inside

PVML.

In Section 6, we then provide experimental results comparing the performance of these four ML algorithms. Our main insight in this section is that the choice of the right ML algorithm used inside PVML depends on two characteristics: first, the ML algorithm's prediction performance (measured in terms of the learning error), and second, our ability to solve the ML-based WDP problem reasonably quickly. We show that the four kernels we test differ along those two dimensions. The Exponential kernel leads to the best prediction performance, but it also results in a computationally very difficult WDP, such that we cannot solve the WDP to optimality when computing the allocation inside the QueryModule. In contrast, the Quadratic kernel only has slightly worse prediction performance, but we can formulate the corresponding WDP as a Quadratic Integer program which we can solve relatively quickly, even for large auction instances. This is why in terms of economic efficiency (our main variable of interest), the Quadratic kernel outperforms the other three kernels.

In Section 7, we then present experimental results evaluating the overall performance of PVML (using SVRs with a Quadratic kernel) and we benchmark it against the CCA. For these experiments, we employ the spectrum auction test suite (SATS) Weiss, Lubin and Seuken (2017), and we use the LSVM, GSVM and MRVM domains. We simulate the CCA with a full clock phase plus an additional supplementary round. The number of bids that bidders submit in the supplementary round is equal to the maximum number of bundle-value pairs that bidders submit in PVML, and we vary that number between 100 and 500.

Our first result is that, as one would expect, the performance of PVML depends on how well the ML algorithm can capture the structure of the domain. LSVM has a highly non-linear structure, but bidders are not interested in a large number of bundles. In this domain, the CCA achieves slightly higher efficiency than PVML with 500 queries. In contrast, in GSVM, the Quadratic kernel can perfectly capture the structure of the domain, and thus is able to generalize very well. This leads to an efficiency of 100% for PVML, even with only 100 queries, while the CCA only achieves 99.2% with 100 queries.

Finally, we test both mechanisms on the MRVM domain, which is the largest and most realistic domain, with 10 bidders and 98 goods. Here, PVML again clearly outperforms the CCA. With 500 queries, the CCA achieves an efficiency of 93.2% while PVML achieves 94.6%, which is roughly a 1.5% point improvement.

1.3. Related Work on Machine Learning and Mechanism Design

There are multiple existing strands of research connecting machine learning with mechanism design. One line of research grew out of the automated mechanism design (AMD) research agenda (Conitzer and Sandholm, 2002, 2004), aiming to use algorithms to design *direct-revelation* mechanisms: these are mechanisms where agents report all of their preferences upfront and the mechanism determines the outcome based on these preferences. The first approaches to AMD were based on formulating the mechanism design problem as a search problem over the space of all possible mappings from agents' preference profiles to outcomes. These approaches were only applicable to small settings mainly because of the dimension of the preference profile space. Recent work has partly addressed this scalability issue by limiting the search to parametrized classes of mechanisms and using learning algorithms to find suitable parameter values. Dütting et al. (2015) used discriminant-based classifiers to learn approximately strategyproof payment rules for combinatorial auctions. Narasimhan, Agarwal and Parkes (2016) further showed how to extend these techniques to mechanisms that are not allowed to charge payments. In more recent work, Duetting et al. (2018) have used deep learning methods to advance the design of revenue-maximizing auctions. Similar deep learning methods were also applied by Golowich, Narasimhan and Parkes (2018) to design matching mechanisms.

The research agenda of using a learning framework to design revenue-maximizing auctions also involves several sample complexity results provided for different settings (Cole and Roughgarden, 2014; Medina and Mohri, 2014; Cesa-Bianchi, Gentile and Mansour, 2015; Morgenstern and Roughgarden, 2015, 2016; Roughgarden and Schrijvers, 2016; Balcan, Sandholm and Vitercik, 2016, 2017; Syrgkanis, 2017). Similarly to AMD,

in our work we design mechanisms through algorithms. However, the main difference lies in our focus of indirect-revelation mechanisms where agents' preferences are iteratively elicited by the mechanism.

Another important research agenda related to this paper is the one of preference elicitation in combinatorial auctions, which dates back to the early 2000s (Sandholm and Boutilier, 2006). Among the most relevant work in this agenda are the papers by Lahaie and Parkes (2004) and Blum et al. (2004) which used learning algorithms to design elicitation algorithms for specific classes of valuations. Elicitation algorithms for generic valuations based on ML were introduced by Lahaie (2011) and further developed by Lahaie and Lubin (2019). These approaches are based on using ML on bidders' reports to find *competitive equilibrium prices*, i.e., prices that coordinate bidders towards demanding efficient allocations. Brero and Lahaie (2018) and Brero, Lahaie and Seuken (2019) built on these mechanisms and designed Bayesian auctions that, besides bidders' reports, can exploit prior beliefs on bidders' valuations to quickly determine competitive equilibrium prices. In contrast to the pre-dominant research agenda in this field, we do not use ML to directly learn or predict competitive equilibrium prices. Instead, we use ML to learn bidders' valuations, and we only an implicit (internal) correspondence to approximate clearing price (but these are never shown to bidders). We use the learned valuations to determine which values should be queried from the bidders in the next round, to ultimately find a highly efficient allocation. This leads to a very different auction protocol, with different opportunities for machine learning, different challenges that need to be addressed, and different resulting properties.

2. Preliminaries

In this section we present our formal model. We start by introducing the combinatorial auction setting in Subsection 2.1. In Subsection 2.2, we discuss the mechanism design problem we want to solve, its challenges, and the solutions proposed in prior work.

2.1. Combinatorial Auctions

In a combinatorial auction (CA), there is a set of m indivisible items being auctioned off to n bidders. We let $[n] = \{1, ..., n\}$, so that $[n]$ and $[m]$ denote the index sets of bidders and items, respectively. A bundle is a subset of the set of items. We associate each bundle with its indicator vector and denote the set of bundles as $\mathcal{X} = \{0, 1\}^m$. We represent the preferences of each bidder i with a value function $v_i : \mathcal{X} \to \mathbb{R}_{\geq 0}$ that is private knowledge of the bidder. Thus, for each bundle $x \in \mathcal{X}, v_i(x)$ represents the *true value* that bidder i has for obtaining x. We do not make any assumptions regarding the structure of bidders' value functions. We assume that values are normalized such that the bidders have zero value for the empty bundle. We sometimes also refer to v_i as bidder i's *valuation*. We let $v = (v_1, \ldots, v_n)$ denote the vector of bidders' valuations and $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ the corresponding vector where bidder i is excluded.

A CA determines an allocation and charges payments to the bidders. An *allocation* is a vector of bundles $a = (a_1, \ldots, a_n)$, with a_i being the bundle that bidder i obtains under a. An allocation is *feasible* if each item is allocated to at most one bidder; we denote the set of feasible allocations by $\mathcal{F}$. *Payments* are defined as a vector $p = (p_1, ..., p_n) \in \mathbb{R}^n$, with p_i denoting the amount charged to bidder i. We assume that bidders have a *quasi-linear utility function* of the form $u_i(a, p) = v_i(a_i) - p_i$. The *social welfare* of an allocation a is the sum of the bidders' true values for a, i.e.: $\sum_{i\in[n]} v_i(a)$. The social welfare-maximizing (i.e., *efficient*) allocation is denoted as $a^*_v \in \arg\max_{a\in\mathcal{F}} \sum_{i\in[n]} v_i(a)$. We measure the efficiency of any allocation $a \in \mathcal{F}$ as $\sum_{i\in[n]} v_i(a) / \sum_{i\in[n]} v_i(a^*_v)$. In this work, we aim for the design of mechanisms that maximize efficiency.

A CA *mechanism* defines how bidders interact with the auction, how the final allocation is determined, and how much each bidder has to pay. In this paper, we design a mechanism that asks bidders to report values for bundles (across different rounds of an auction). We let $\hat{v}_{i,x}$ denote bidder i's value report on bundle x; we often refer to $(x, \hat{v}_{i,x})$ as a *bundle-value pair*. Our mechanism will enforce that a bidder cannot make multiple

reports on the same bundle. We let s_i denote a *set of bundle-value pairs* reported by bidder i. For notational simplicity, we say that $x \in s_i$ if $(x, \hat{v}_{i,x}) \in s_i$. We use $|s_i|$ to denote the number of bundle-value pairs in s_i.

In general, bidders do not make a value report for all bundles in the bundle space. Therefore, given a set of reported bundle-value pairs by bidder i, s_i, we define a *reported value function* $\hat{v}_{s_i}(x)$ which monotonically extends bidder i's individual value reports $\hat{v}_{i,x}$ for bundles $x \in s_i$ to the whole bundle space assuming free disposal (i.e., $\hat{v}_{s_i}(x) = \max\{\hat{v}_{i,x'} : x' \subseteq x \text{ and } (x', \hat{v}_{i,x'}) \in s_i\}$). Thus, formally, every set of bundle-value pairs s_i induces a different function $\hat{v}_{s_i}(x)$. To avoid the double subscript notation, going forward, we simply write $\hat{v}_{i,s}(x)$ for $\hat{v}_{s_i}(x)$.

We let $s = (s_1, \ldots, s_n)$ denote the vector of these sets of reported bundle value pairs. Given an s, we can now define the *reported social welfare* of an allocation a as $\sum_{i\in[n]} \hat{v}_{i,s}(a_i)$. The *efficient allocation with respect to* s is denoted as $a^*_{\hat{v}_s} \in \arg\max_{a\in\mathcal{F}} \sum_{i\in[n]} \hat{v}_{i,s}(a_i)$. Without loss of generality, we only consider allocations $a^*_{\hat{v}_s}$ that consist of bundles contained in s.

2.2. The Vickrey-Clarke-Groves (VCG) Mechanism

We now define the well-known VCG mechanism (Vickrey, 1961; Clarke, 1971; Groves, 1973) which is the gold standard to allocate items efficiently in a CA.

Definition 1 (VCG Mechanism)**.** *Under VCG, every bidder i must make a report about its full value function which we denote by $\hat{v}_i$. VCG is then defined by:*

- *An allocation rule which allocates items according to $a^*_{\hat{v}} \in \arg\max_a \sum_i \hat{v}_i(a_i)$*
- *A payment rule which charges each bidder i a payment equal to:*

$$p_i^{VCG} = \sum_{j\neq i} \hat{v}_j(a^*_{\hat{v}_{-i}}) - \sum_{j\neq i} \hat{v}_j(a^*_{\hat{v}})\,. \tag{1}$$

It is well known that VCG is *strategyproof*, i.e., it is a dominant strategy for all bidders to report their true value function, independent of what the other bidders do. This can easily be seen by considering bidder i's utility under VCG:

$$u_i(a^*_{\hat{v}}, p_i^{VCG}) = \left(v_i(a^*_{\hat{v}}) + \sum_{j\neq i} \hat{v}_j(a^*_{\hat{v}})\right) - \sum_{j\neq i} \hat{v}_j(a^*_{\hat{v}_{-i}}) \tag{2}$$

Bidder i cannot affect the last term $\sum_{j\neq i} \hat{v}_j(a^*_{\hat{v}_{-i}})$, and the sum of the first two terms is equal to reported social welfare, which is maximized by the mechanism. Thus, it is dominant strategy for bidder i to report his true value function. When all bidders follow their dominant strategy, VCG is efficient.

2.3. Iterative VCG Mechanisms

VCG is often considered an impractical mechanism for CAs, in particular in domains with a large number of items, because it requires bidders to report their full value function (which is exponentially sized in the number of items). Fortunately, it has been shown that one does not always need to elicit the full value functions to determine the VCG outcome (Mishra and Parkes, 2007). Building on this insight, researchers have proposed *iterative VCG mechanisms* (e.g., Mishra and Parkes (2007); de Vries, Schummer and Vohra (2007)). The idea of these mechanisms is to interact with bidders over multiple rounds and, when bidders follow truthful bidding behavior (suitably defined), to only elicit enough information to determine the VCG outcome with respect to their true valuation at the end of the auction.

Intuitively, by realizing the VCG outcome, these mechanisms inherit some good incentive properties of VCG. In fact, such mechanisms support truthful bidding in ex-post Nash equilibrium (Mishra and Parkes, 2007; de Vries, Schummer and Vohra, 2007). However, given that iterative VCG mechanisms increase the strategy space (where bidders may now condition their actions on progress throughout the auction), we lose the strong guarantees of VCG based on dominant strategies.

3. The Pseudo VCG Machine Learning-based Mechanism

In this section, we introduce our new mechanism. As mentioned in the previous section, our primary design goal is to achieve high efficiency. Unfortunately, it is well known that finding the efficient allocation requires communication that is exponential in the number of items in the worst case (Nisan and Segal, 2006). Furthermore, it has been shown that you cannot even achieve a useful approximation of efficiency with less than exponential worst case communication. Note that this implies that iterative VCG mechanisms (e.g., Mishra and Parkes (2007); de Vries, Schummer and Vohra (2007)) also require exponential communication in the worst case and are thus impractical even in medium-sized domains.

To obtain a practical CA, we design a mechanism that, like iterative VCG, interacts with bidders over multiple rounds, but that imposes a *strict limit* on the information exchanged between each bidder and the auction. Note that the amount and the type of information being exchanged depends on the auction format and thus the type of queries that are used. Many iterative CAs use *demand queries* where bidders are shown ask prices and have to report their profit-maximizing bundle at these prices. The amount of information exchanged per demand query depends on the dimensionality of the ask prices. For example, the CCA uses linear prices (i.e., one price per item), which, combined with a bound on the number of rounds, ensures practicality. Note that, due to the result by Nisan and Segal (2006), this also implies that the CCA cannot provide efficiency guarantees.

In our mechanism, the information exchanged consists of a small set of bundle-value pairs from each bidder, and the size of this set is limited via a parameter of our mechanism. Of course, due to Nisan and Segal (2006), this implies that our mechanism also cannot provide useful efficiency *guarantees*. However, our goal is to design a mechanism that maximizes *empirical efficiency* in realistic CA domains.

Intuitively, a mechanism that wants to maximize empirical efficiency but is constrained on information exchange should elicit the "most important" information from each bidder. However, in an iterative mechanism, identifying the "next most-useful query" is not an easy task because it requires reasoning about the incremental value of the missing information given what has already been elicited. To this end, there are three basic approaches. First, if enough *structure* regarding bidders' value functions can be assumed ex ante, then the auctioneer can use this structure to *generalize* one piece of information from bidders to a large part of the bidders' value functions. However, in this work, we allow for general valuations such that our mechanism does not rely on any critical structural assumptions.

Second, instead of the auctioneer performing the generalization task by exploiting structural assumptions, the bidders can do it themselves by explicitly providing information statements about their whole value function. The CCA follows this principle by asking bidders demand queries in each round of the clock phase. An answer to a demand query does not only provide the auctioneer with information on one bundle, but it is simultaneously a statement regarding the bidder's whole value function. The auctioneer can then use these "global information statements" to identify the next query. For example, the CCA computes the next query via a simple price update rule where prices on over-demanded items are increased by a fixed percentage (Ausubel and Baranov, 2017).

In this work, we propose a third approach to perform the generalization task, using *machine learning (ML)*. The idea is to train an ML algorithm for each individual bidder on that bidder's value reports to *generalize* to

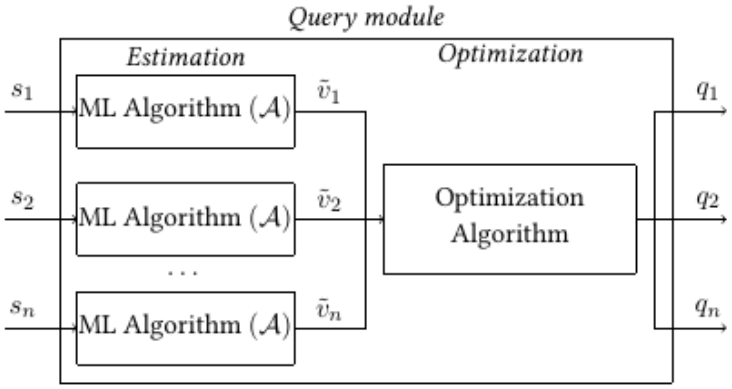


Figure 1: Schematic representation of how the query module works.

the whole bundle space, i.e., to *infer* a value for each bundle the bidder has not evaluated yet. For example, consider a bidder who reported value 1 for bundle $(1, 0)$ and value 10 for bundle $(1, 1)$. To build some intuition, consider simple *linear regression*, which in this example would predict value 9 for bundle $(0, 1)$. Of course, linear regression can only learn additive value functions as it only has one coefficient per item. In Section 5, we present our general ML framework and explain how non-linear ML algorithms (which can capture substitute and complements valuations) can be employed in our mechanism. Equipped with an ML algorithm, we can take the bundle-value reports obtained from each bidder, feed them to the ML algorithm to generalize to the whole bundle space. As in approaches 1 and 2, this provides the auctioneer with a statement about the whole value function which can then be used to compute the next query. In Section 3.1, we describe in detail our machine learning-powered "QueryModule" which takes as input a set of bundle-value pairs from each bidder and determines a new query for each bidder.

In Section 3.2 we present our full ML-powered mechanism which uses the QueryModule as a sub-routine in every round of the auction, and which ultimately determines the final allocation and payments. Recall that, due to the result by Nisan and Segal (2006), we cannot provide efficiency guarantees while maintaining practicality, which implies that we cannot determine the VCG outcome. While this implies that we cannot achieve the good efficiency and incentive guarantees of iterative VCG mechanisms, we nevertheless follow the *design principles of VCG* to obtain a mechanism that provides bidders with "good incentives in practice." Concretely, we charge "VCG-style payments" and we allow bidders to "push" information to the auction which they deem useful for efficiency. Together, these two design features lead to good incentives in practice, as we will show in detail in Section 4.

3.1. Machine Learning-powered Query Module

In this section, we describe in detail our QueryModule (see Figure 1 for a schematic view). The module takes as input a set of bundle-value pairs s_i from each bidder i. Then, for each bidder i a ML algorithm $\mathcal{A}$ is trained based s_i to infer a value function $\tilde{v}_i$.[1] The inferred value functions are then used by the *optimization algorithm* to compute an efficient allocation with respect to the inferred valuations. The module assigns each bidder i a new query q_i given by the bundle she obtains under this allocation.[2]

[1] In Section 5, we first instantiate $\mathcal{A}$ using linear regression and then generalize it to non-linear learning models.

[2] To prevent a bidder from being assigned a bundle she has already evaluated, we restrict the set of allocations considered when determining a new query for each bidder.

Algorithm 1: Machine Learning-powered Query Module

1 function NextQuery$(\mathcal{A}, s)$;
Inputs: ML algorithm $\mathcal{A}$; Vector of sets of bundle-value pairs $s = (s_1, ..., s_k)$;
2 **foreach** bidder $i \in [k]$ **do**
3 $\quad \tilde{v}_i = \mathcal{A}(s_i)$; $\quad$ **Estimation Step:** infer valuation for each bidder using ML algorithm
4 **end**
5 Determine $\tilde{a} \in \arg\max_{a \in \mathcal{F}} \sum_{i \in [k]} \tilde{v}_i(a)$; $\quad$ **Optimization Step** (based on inferred valuations)
6 **foreach** bidder $i \in [k]$ **do**
7 $\quad$ **if** $\tilde{a}_i \notin s_i$ **then**
8 $\quad\quad q_i = \tilde{a}_i$;
9 $\quad$ **else**
10 $\quad\quad \mathcal{F}_i = \{a \in \mathcal{F} : \forall x \in s_i,\ a_i \neq x\}$;
11 $\quad\quad$ Determine $\tilde{a}' \in \arg\max_{a \in \mathcal{F}_i} \sum_{i \in [k]} \tilde{v}_i(a)$; $\quad$ **Optimization Step** (with restrictions)
12 $\quad\quad q_i = \tilde{a}'_i$;
13 $\quad$ **end**
14 **end**
15 Output vector of queries $q = (q_1, ..., q_k)$;

Algorithm 1 provides a detailed description of our query module. Formally, we represent our query module as a function NextQuery$(\mathcal{A}, s)$ which takes as input a machine learning algorithm $\mathcal{A}$ and a set of bundle-value pairs s_i from each bidder i. As we will describe in Subsection 3.2, our mechanism calls our query module on different subgroups of bidders called *economies*. To emphasize this aspect, in Algorithm 1 we denote the set of bidders as $[k]$ (instead of $[n]$). Our algorithm begins with the "Estimation Step": for each bidder $i \in [k]$ the ML algorithm $\mathcal{A}$ is trained based on the bundle-value pairs s_i to infer a value function $\tilde{v}_i$ (Line 3). Then, for each bidder $i \in [k]$ our Algorithm determines a set $\mathcal{F}_i$ consisting of those feasible allocations that do not assign bidder i a bundle she has already evaluated in s_i (Line 6). In the "Optimization Step" an optimization algorithm is evoked to determine the allocation $\tilde{a}$ in $\mathcal{F}_i$ achieving the highest social welfare according to $\tilde{v}$ (Line 7). The algorithm assigns each bidder i the new query $q_i = \tilde{a}_i$ (Line 8). Once a new query has been assigned to each bidder i in $[k]$, Algorithm 1 outputs the vector of queries q.

3.2. The PVML Mechanism

In this subsection we describe our PVML mechanism. PVML uses the query module introduced in Subsection 3.1 to determine a highly efficient allocation and charge "VCG style payments". These payments are obtained by charging each bidder i the difference between the overall value realized by the other bidders in an allocation elicited excluding bidder i and the one they realize in the final allocation. We call the group of all bidders other than i *bidder i's marginal economy* – which differentiates from the *main economy* consisting of all bidders.[3] As we will see in Section 4, charging these VCG style payments while allowing bidders to submit self chosen bundle-value pairs motivates bidders to report true values throughout the auction.

Algorithm 2 provides a formal description of PVML. The mechanism is parametrized by a machine learning algorithm $\mathcal{A}$ that will be given as input to the function *NextQuery*, a maximum number $Q_{\max}$ of value reports that it is allowed to ask each bidder, and a maximum number Q_0 of value reports that can be selected by a bidder in the first stage of the auction. When the auction begins, the mechanism asks each bidder i to report

[3]The idea of employing multiple elicitation runs in main and marginal economies in an auction was firstly proposed by Ausubel (2006) to design an iterative VCG mechanism that allocates multiple heterogeneous commodities.

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 $t = t + 1$;

8 Generate queries for the main economy: *NextQuery*$(\mathcal{A}, s^{t-1})$;

9 Generate queries for each bidder i's marginal economy: *NextQuery*$(\mathcal{A}, s^{t-1,(-i)})$;

10 Send generated queries to bidders and ask for corresponding values;

11 Let s' denote all reported bundle-value pairs obtained in Step 10 and let $s^t = s^{t-1} \cup s'$;

12 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};

$Q_i^{push} \leq Q_0$ self-chosen bundle-value pairs (Line 1) and obtains the remaining $Q_0 - Q_i^{push}$ reports by asking her values for randomly-selected bundles (Line 2). The set of bundle value pairs obtained from each bidder i is denoted as s_i^0. PVML uses these bundle-value pairs to initialize $n + 1$ vector of reports, one related to the main economy (Line 3) and one for each marginal economy (Line 4). Then, while the mechanism is allowed to ask up to n queries to any bidder without violating the query budget Q_{max} , the following procedure is iterated: First, the mechanism invokes the function NextQuery on the value reports corresponding to the main economy (Line 8) and on the ones corresponding to each marginal economy (Line 9).[4] All the queries generated in Line 8 and 9 are sent to the bidders that are asked to report the corresponding values (Line 10). The new vector of reports obtained is integrated in the set of queries corresponding to the main economy (Line 11). Each vector or reports corresponding to the queries generated for a marginal economy in Line 9 is integrated in the reports for this economy (Line 12). When the iterations stop, PVML determines its final allocation and payments: The final allocation a^{pvml} is the value-maximizing allocation given all the reports obtained in the elicitation for the main economy (Line 14). The payment p_i^{pvml} charged to each bidder i is given by the difference between the overall value realized in the optimal allocation for the values elicited in his marginal economy and the one the other bidders realize in a^{pvml} (Line 15).

Remark 1 (Answering Value Queries with Upper and Lower bounds)**.** *Note that in PVML bidders need to*

[4]Note that two different calls of function NextQuery may assign the same query to the same bidder. Thus, the overall number of queries assigned in Line 8 and 9 may not be the same across bidders. To let PVML assign each bidder the same number of queries at each round, we equalize queries by iteratively calling the function NextQuery on the vector of reports related to the main economy, discarding those queries assigned to bidders that have already been assigned the maximum number of queries.

report exact values for the bundles queried in Line 10 of Algorithm 2. However, in practice it may be easier for bidders to specify bounds on bundle values than to determine such values precisely (see, e.g., Parkes (2006)). In Brero, Lubin and Seuken (2018) we have introduced a modified machine learning-based elicitation algorithm that works when bidders report upper and lower bounds instead of exact values. Via simulations we have shown that our ML-powered elicitation algorithm determines highly efficient allocations even under these less informative reports. A new PVML version based on this algorithm can be designed in future work.

4. Theoretical Analysis

In this section, we analyze which theoretical properties PVML satisfies.

4.1. Learning Error and Elicitation Guarantees

The *prediction performance* of an ML algorithm is usually assessed via its *learning error*. Formally, for a bidder i, we define the learning error in a bundle x as the absolute value of the difference between the bidder's true value and the inferred value for the bundle x, i.e.: $\text{err}(x) = |v_i(x) - \tilde{v}_i(x)|$. Of course, in practice, an auctioneer cannot evaluate the learning error because she does not have access to the true valuations. However, she can evaluate the ML algorithm using prior auction data. In general, the better the ML algorithm captures the domain, the smaller the learning errors can be expected. In Section 6, we evaluate the learning error of four different ML algorithms and show how they depend on the richness of the ML model used.

In this section, we still leave the ML algorithm unspecificied to derive theoretical properties that hold for *any* ML algorithm. We show how the performance of the ML algorithm, measured in terms of the learning error, relates to the performance of PVML. Specifically, given a bound on the learning error, we first derive a bound on the efficiency of PMVL, and then we show how we can impute prices to obtain an approximate competitive equilibrium.

4.1.1. Bounding the Efficiency Loss

First we use the learning error to bound the efficiency loss of PVML. Consider the QueryModule described in Algorithm 1. In Line 5, we compute $a^*_{\tilde{v}}$, i.e., the efficient allocation with respect to the inferred valuation profile $\tilde{v}$. We then use this allocation in Line 8 to assign a new query for each bidder (unless the bidder has already reported a value for the corresponding bundle). Thus, at the end of the QueryModule, PVML ensures that all bundles contained in $a^*_{\tilde{v}}$ have been queried and will ultimately be used to compute the final allocation. This implies that the efficiency of PVML can only only be (weakly) higher than the efficiency of $a^*_{\tilde{v}}$, assuming that all bidders only make truthful bundle-value reports. Thus, a bound on the efficiency loss for any specific $a^*_{\tilde{v}}$ translates into a bound on the efficiency loss of PVML.

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:*

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

Proof. We add and subtract $\sum_{i\in[n]} v_i(a^*_{\tilde{v}})$ and $\sum_{i\in[n]} \tilde{v}_i(a^*_v)$ to $\sum_{i\in[n]} v_i(a^*_v) - v_i(a^*_{\tilde{v}})$, obtaining

$$\sum_{i\in[n]} v_i(a^*_v) - v_i(a^*_{\tilde{v}}) = \left(\sum_{i\in[n]} v_i(a^*_v) - \tilde{v}_i(a^*_v)\right) + \left(\sum_{i\in[n]} \tilde{v}_i(a^*_{\tilde{v}}) - v_i(a^*_{\tilde{v}})\right) + \left(\sum_{i\in[n]} \tilde{v}_i(a^*_v) - \tilde{v}_i(a^*_{\tilde{v}})\right). \quad (5)$$

Given that $a^*_{\tilde{v}}$ is a social welfare-maximizing allocation under $\tilde{v}$, the term $\sum_{i\in[n]} \tilde{v}_i(a^*_v) - \tilde{v}_i(a^*_{\tilde{v}})$ in (5) cannot be positive. Inequality (4) follows by considering that any real number cannot be greater than its absolute value. □

In words, Proposition 1 shows us that the efficiency loss in $a^*_{\tilde{v}}$ is bounded by the overall learning error in the two allocations a^*_v and $a^*_{\tilde{v}}$ (normalized by the maximum social welfare). While the efficient allocation a^*_v is not known to the auctioneer, she does know $a^*_{\tilde{v}}$ at each round of PVML. This provides some motivation for the design of our QueryModule. Consider the allocation $\tilde{a}$ which Algorithm 1 determines in Line 5. The bound from Proposition 1 increases in the learning errors in each bidder's bundle $\tilde{a}_i$. This motivates Step 8 of Algorithm 1, where we query every bidder the bundle $\tilde{a}_i$; if the allocation remained stable in the next round, there are no more learning errors due to $\tilde{a}$ that contribute to the bound (when using an ML algorithm that is accurate on the training data). If a bidder i has previously reported a value for bundle $\tilde{a}_i$, then this means that there are no "obvious" queries to ask this bidder. Consequently, Algorithm 1 then enters the "else-branch" to determine a new allocation where the winner determination algorithm is restricted to only allocate a bundle to this bidder which he has not reported a value for yet.

4.1.2. Learning Error and Imputed Approximate Clearing Prices

Recall that in PVML, bidders submit bundle-value reports, while most prior work on iterative CAs uses prices for elicitation. We now show how we can obtain a price-based interpretation of the elicitation performed by PVML's QueryModule. In particular, we show how we can impute prices that are implicit in the elicitation. Given a bound on the learning error of the ML algorithm, we then provide a measure on the quality of these imputed prices. Concretely, we measure the quality of these prices by using the concept of competitive equilibrium.

To define a competitive equilibrium, we introduce a very general concept of *prices* (allowing for non-anonymous bundle prices). We let $\pi = (\pi_1, .., \pi_n)$ denote the *price profile*, where each π_i is bidder i's *price function*, with $\pi_i(x)$ denoting bidder i's price for any given bundle $x \in \mathcal{X}$. Given this, we can now define a competitive equilibrium (CE).

Definition 2 (Competitive equilibrium)**.** *Given prices π, we define each bidder i's demand set d^π_i as the set of bundles that maximize her utility at π: $d^\pi_i = \arg\max_{x\in\mathcal{X}} v_i(x) - \pi_i(x)$. 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, $a_i \in d^\pi_i$.*

It is well known that any competitive equilibrium allocation is efficient (Mas-Colell, Whinston and Green, 1995, 16.C-D). Prior work on iterative CAs has exploited this property. For example, the iterative VCG mechanisms by Mishra and Parkes (2007) and de Vries, Schummer and Vohra (2007) iteratively update prices until a CE is found.[5] This approach is motivated by the fact that any auction that finds an efficient allocation must

[5] More concretely, they find specific CE prices called *universal competitive equilibrium (UCE) prices* which contain all the information necessary to compute a VCG outcome.

reveal CE prices (Nisan and Segal, 2006). However, in the worst case, this may require an exponential amount of communication (potentially quoting a different price for each bundle).

Recall that in PVML, we limit the amount of information exchanged (via the query cap). Thus, due to the result by Nisan and Segal (2006), we cannot guarantee finding a CE. In fact, PVML does not even use prices to interact with bidders (in contrast to the CCA or iterative VCG mechanisms). However, perhaps surprisingly, at each round of the auction, we can *impute* prices based on the inferred valuations in that round. This provides insight into how bundles are being *implicitly priced* in each round of PVML.

Concretely, if $\tilde{v}$ is the inferred valuation profile with respect to all bundle-value pairs reported so far, then one can *impute* a price profile $\pi = \tilde{v}$ for the current round.[6] To measure the quality of this imputed price profile, we use the concept of approximate competitive equilibrium prices (see, e.g., Brero and Lahaie (2018)). Specifically, given a bound on the learning error of the ML algorithm, we can bound the degree of approximation in the equilibrium price profile.

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.*

Proof. The transfer that should be made to the seller to trade a^*_v at prices $\tilde{v}$ is $\sum_{i\in[n]} \tilde{v}_i(a^*_{\tilde{v}}) - \sum_{i\in[n]} \tilde{v}_i(a^*_v)$, while the one that should be made to each bidder i is $\big(\max_{x\in\mathcal{X}} v_i(x) - \tilde{v}_i(x)\big) - \big(v_i(a^*_v) - \tilde{v}_i(a^*_v)\big)$. After summing up all these transfers we obtain a total transfer

$$\delta = \left(\sum_{i\in[n]} \max_{x\in\mathcal{X}} v_i(x) - \tilde{v}_i(x)\right) + \left(\sum_{i\in[n]} \tilde{v}_i(a^*_{\tilde{v}}) - v_i(a^*_v)\right). \tag{6}$$

Note that, by adding and subtracting $\sum_{i\in[n]} \tilde{v}_i(a^*_v)$ to the second term in Equation (6) we have that $\sum_{i\in[n]} \tilde{v}_i(a^*_{\tilde{v}}) - \sum_{i\in[n]} v_i(a^*_v) \leq \sum_{i\in[n]} \tilde{v}_i(a^*_v) - v_i(a^*_v)$. We can then use our assumptions to conclude that $\delta \leq n(\delta_1+\delta_2)$. □

Proposition 2 provides us with a measure on the quality of the price profile $\pi = \tilde{v}$. Concretely, it tells us how close to clearing this price vector is as a function of the learning error of the ML algorithm.[7] We want to emphasize that these imputed prices are only implicit when PVML is run. Importantly, they inherit the structure of $\tilde{v}$, which can be very complex, depending on the ML algorithm used. In particular, the prices will in general be non-anonymous, and if we use high-dimensional (non-linear) ML algorithms, then they will also be high-dimension bundle prices. Note that this implict high-dimensional price structure enables PVML to find an approximate CE (where the approximation depends on the learning error of the ML algorithm) while approaches based on linear prices may severely limited.

Introducing this price-based interpretation of our QueryModule allows us to explain how our mechanism is related to the work by Lahaie and Parkes (2004) who were among the first to establish a theoretical connection between preference elicitation in CAs and machine learning. In this prior work, they proposed an iterative ML-based elicitation paradigm which also uses inferred valuations to drive the elicitation process. However, while PVML is based on value reports, their algorithm critically requires demand queries (i.e., it communicates ask prices to bidders in every round). The advantage of their approach is that it guarantees finding a competitive equilibrium. However, due to the result by Nisan and Segal (2006), if their approach

[6] Of course, $\pi = \tilde{v}$ is the seller-optimal imputed price vector. But it is not the only price vector one can impute from $\tilde{v}$. Instead, there is a polytope of imputed prices for which the following proposition holds. We will add more details on this in a future version of this paper.

[7] Note that, starting from the approximately-clearing price profile $\pi = \tilde{v}$, we could now derive a bound on the efficiency loss in $a_{\tilde{v}}$. However, it would be the same bound we have already proven in Proposition 1.

was applied in a general setting, it would require communicating exponentially-sized prices to the agents in every round, which makes it impractical in such settings.

In contrast to the approach by Lahaie and Parkes (2004), the CCA is designed with practical applications in mind (like PVML). A guiding pricinple underlying the clock phase of the CCA is that it aims to find approximate CE prices. However, as a practical mechanism, it cannot use exponential communication; instead, it uses linear prices and a limited number of rounds. Thus, due to Nisan and Segal (2006), it cannot guarantee finding a true CE. Note that the supplementary round of the CCA (allowing bidders to submit up to 500 bundle-value pairs in one final round) is designed to address the potential inefficiencs that remain at the end of the clock phase (e.g., due to using linear prices that may be out of equilibrium). Note that both, PVML and CCA are practical auction designs, but they restrict the amount of information exchanged in different ways. Furthermore, while the CCA explicitly aims to find an approximate CE by using demand queries, PVML can be interpreted as doing so implicitly by using an ML algorithm on value reports.

4.2. Incentives

We now analyze the incentive properties of PVML. Specifically, we show how charging the VCG style payments of Equation 3 while allowing bidders to push some bundle-value pairs at the beginning of the auction motivates bidders to follow *truthful strategies*, which we define as follows:

Definition 3 (Truthful Strategy)**.** *We say that a bidder is* truthful *under PVML if, when a bidder i adopts this strategy during the auction, he*

- *always reports his true values for the bundles explicitly queried by PVML, and*
- *he only pushes bundle-value pairs $(x, \hat{v}_{ix})$ with $\hat{v}_{ix} = v_i(x)$.*

As described in Subsection 4.1.1, PVML is only guaranteed to find highly efficient allocations when bidders report their values truthfully (assuming that the ML algorithm $\mathcal{A}$ accurately captures bidders' valuations). Thus, we want to provider bidders with good incentives to follow a truthful strategy. Unfortunately, as shown by the following proposition, PVML allows a bidder to increase her final utility by misreporting some of her values, even when the other bidders are truthful.

Proposition 3. *Under PVML, a bidder may benefit from misreporting her values, even when the other bidders are truthful.*

Proof. To show this result, we need to carefully execute one run of PVML (over multiple rounds) for one specific auction instance. Consider a setting with 2 bidders and 2 items. We use PVML with linear regression, which in this setting estimates 2 coefficients (one per item). The PVML parameters are set as follows: $Q_{\max} = 3$ and $Q_0 = 1$. Bidder 1's true values are $v_1(1,0) = v_1(1,1) = 2$, $v_1(0,1) = 1.1$, and bidder 2's ones are $v_2(1,0) = v_2(0,1) = 1$, $v_2(1,1) = 2$.

Implicitly, PVML assumes a report from both of zero for the empty bundle. Additionally, assume that in Line 1 of Algorithm 2, bidder 2 truthfully reports $\hat{v}_{2,(1,1)} = 2$. Assume that in Line 1 bidder 1 truthfully reports $\hat{v}_{1,(0,1)} = 1.1$. After the initialization in Lines 3 and 4, we have: $s_1^0 = s_1^{0,(-2)} = \{((0,1), 1.1)\}$, $s_2^0 = s_2^{0,(-1)} = \{((1,1), 2)\}$. The valuations inferred in the Estimation Step of the query module are: $\tilde{v}_1(1,0) = 0$ and $\tilde{v}_1(0,1) = \tilde{v}_1(1,1) = 1.1$, and $\tilde{v}_2(1,0) = \tilde{v}_2(0,1) = 1$, and $\tilde{v}_2(1,1) = 2$. Under this inference, bidder 1 is assigned query $(1,1)$ in both the main and bidder 2's marginal economy, while bidder 2 is assigned query $(1,0)$ in both the main and bidder 1's marginal economy (assuming ties are broken in favor of the first item). If bidders are truthful, in Line 11 the two value reports $\hat{v}_{1,(1,1)} = 2$ and $\hat{v}_{2,(1,0)} = 1$ are included in the corresponding sets of bundle-value pairs. When the auction ends, bidder 1 is allocated bundle $(0,1)$ and charged $p_1^{pvml} = 1$, with utility equal to $1.1 - 1 = 0.1$.

If bidder 1 had reported $\hat{v}_{1,(0,1)} = 0.9$, the valuation $\tilde{v}_1$ inferred in the Estimation Step of the query module would have been $\tilde{v}_1(1,0) = 0$ and $\tilde{v}_1(0,1) = \tilde{v}_1(1,1) = 0.9$, and he would have been assigned query $(1,0)$ in the main economy, obtaining this bundle in the final allocation for a payment equal to 1. Then, his utility would have been $2 - 1 = 1 > 0.1$. □

Remark 2 (Robustness Against Strategic Manipulation in Practice)**.** *While PVML is not robust against manipulations, we note that finding a beneficial manipulation of the kind described above may be very difficult for bidders. Indeed, if we look again at the proof, we notice that bidder 1 can only successfully perform his manipulation because he has perfect information about the current state of the auction (all reported bundles and the inferred valuations) and because he knows which values bidder 2 is going to report. To make PVML more robust to strategic manipulations in practice, we suggest prohibiting bidders from communicating with each others during the auction (as is common in practical auction designs). Furthermore, in Step 10 of the PVML mechanism, we suggest sending the list of new queries to each bidder as an un-ordered list of queries, such that a bidder cannot deduce which queries belong to the main or marginal economies, respectively. We argue that, from a practical perspective, this makes it even harder for a bidder to find a beneficial manipulation because it is now very difficult for a bidder to estimate the impact of any manipulation. This idea is similar to the one presented in Parkes (2001) and, as discussed by Parkes (2001), it can also hinder collusion or spiteful behaviors.*

Note that allowing bidders to push bundle-value pairs in the initial round of PVML makes the manipulation described in the proof of Proposition 3 unnatural. Instead of misreporting his value for bundle $(0,1)$ to ultimately be queried and allocated bundle $(1,0)$, bidder 1 could just push bundle-value pair $(1,0)$ in the initial round, and then report truthfully afterwards. In fact, this ability to push bundle-value pairs is an important design feature of PVML, as it makes many manipulations unnatural. In fact, we argue that this design feature of PVML togehter with the VCG-style payment rule, leads to the good incentives in practice. To make this notation more formal, we now first introduce some new notation.

We denote as σ_i a generic strategy for bidder i, determining the behavior of the bidder at any point throughout the auction (which of course depends on his valuation and any other information available to the bidder). We let σ denote the bidders' strategy profile. We use τ_i to denote a *truthful strategy*, i.e., a strategy where bidder i always reports truthful values, and we let τ denote the corresponding truthful strategy profile. A CA mechanism f is defined as follows: $f(\sigma) = \langle g(\sigma), h(\sigma)\rangle$ over strategy profiles σ, where $g(\sigma)$ outputs the allocation, and $h(\sigma)$ outputs the payment profile. With this we can now define the following property.

Definition 4 (Social Welfare Alignment)**.** *A mechanism $f = \langle g, h\rangle$ is* social welfare aligned *if, for all bidders $i \in [n]$, given a truthful strategy profile $\tau = (\tau_i, \tau_{-i})$, for any strategy $\sigma_i \neq \tau_i$, the following holds for some $\gamma > 0$:*

$$\Big(u_i(f(\sigma_i, \tau_{-i})) = u_i(f(\tau)) + \gamma\Big) \Leftrightarrow \Big(V(g(\sigma_i, \tau_{-i})) = V(g(\tau)) + \gamma\Big)$$

The following result is now straightforward.

Proposition 4. *PVML is social welfare aligned.*

Proof. Given a PVML mechanism $f = \langle g, h\rangle$, by plugging in the payment formulation of PVML, the utility of each bidder i, when the other bidders follow a truthful strategy profile τ_{-i}, can be written as

$$u_i(f(\sigma_i, \tau_{-i})) = v_i(a^{pvml}) + \sum_{j \neq i} v_j(a^{pvml}) - \sum_{j \neq i} v_j^{(-i)}(a^{(-i)}) = V(a^{pvml}) - \sum_{j \neq i} v_j^{(-i)}(a^{(-i)}), \qquad (7)$$

where $a^{pvml} = g(\sigma_i, \tau_{-i})$. Note that when the other bidders are truthful, bidder i cannot affect $a^{(-i)}$. Thus, any strategy that increases bidder i's utility by some amount γ must increase the social welfare by the same

amount γ, and vice versa, which shows that PVML is social welfare aligned.[8] □

Discussion. Now that we have shown that PVML satisfies social welfare alignment, we have all pieces in place to argue that PVML provides bidders with good incentives in practice. By "good incentives in practice" we mean that a bidder should follow the following simple strategy: (a) in the initial phase, push bundles you are interested in winning and that you think you have a chance of winning, (b) only make truthful value reports. If the query cap for the initial round Q^{max} is sufficiently large (e.g., 50-100), then in practice, bidders would have a lot of opportunity to provide (push) their initial knowledge to the auctioneer.

In this argument, we take as a baseline the benchmark where PVML simply queries bidders randomly in the initial round and where bidders answer all queries truthfully. There are two cases to consider. First, the simple case, when a bidder does not have any knowledge or belief that he can manipulate in a way such as to increase the efficiency of the final allocation compare to the baseline. In this case, it follows from the fact that PVML is social welfare aligned that the bidder is best off to simply report truthfully.

For the second case, assume the bidder has knowledge (or a belief) about which allocations could lead to a better social welfare than the baseline. Then the bidder could try to execute a sophisticated manipulation (with the goal of getting PVML to eventually implement that allocation). It would be better if the bidder could directly push this allocation to the auctioneer and be truthful afterwards. However, in practice, it is unlikely that bidders will be able to identify (or form beliefs about) about complete allocation (which requires an assignment of one bundle per bidder) with higher social welfare than the one PVML would determine by default, while it is more reasonable to assume that bidders can form beliefs regarding which bundles they will likely receive *as part* of highly efficient allocations. Thus, by letting bidder push those bundles to the mechanism, the mechanism can do the job of figuring out the remaining piece of the allocation (and it is designed to do this well). This provides the bidder with the incentive to push those bundles and report truthfully afterwards.

13. REMARK (again practical PVM):

4.3. Individual Rationality

A mechanism is *individually rational* if each bidder's payment is less than or equal to the bidder's reported value for his final allocation. We can show:

Proposition 5. *PVML satisfies individual rationality.*

Proof. *We show that, for each bidder i, $\hat{v}_i(a^{pvml}) - p_i^{pvml} \geq 0$. From the definition of the PVM payment rule this is equivalent to $\sum_{j\in[n]} \hat{v}_j(a^{pvml}) - \sum_{j\neq i} \hat{v}_j(a^{(-i)}) \geq 0$. This holds because a^{pvml} is optimized over the query profile $\hat{v}^\star$ that includes $\hat{v}^{(-i)}$ used to compute $a^{(-i)}$.*

4.4. No-Deficit

Ideally, a mechanism stipulating transfers between its participants and the center should not run a *deficit*. However, PVML (as defined in Section 3) does not guarantee the *no deficit* property. From Equation (3) we note that each payment p_i^{pvml} is expected to be positive because the allocation $a^{(-i)}$ is derived as an optimal allocation for the marginal economy excluding bidder i. But it may be the case that the allocation a^{pvml} happens to achieve higher reported social welfare with respect to the setting excluding bidder i than $a^{(-i)}$. When this happens, our mechanism may run a deficit.

[8] Note social welfare alignment is a property that is satisfied, not only by PVML, but by a large class of mechanisms. For example, all *VCG-based mechanisms* as introduced by Nisan and Ronen (2007) are social welfare aligned.

Although a deficit violation is formally possible in PVML, we did not observe observe *any* in the experiments we describe in Section 7 (when sufficiently many queries are allowed). However, for a practical implementation of PVML, most auctioneers would likely want a guarantee that the mechanism does not run a deficit. To obtain this guarantee we can simply lower bound each bidder's payment by zero (or by a reserve price), i.e., we charge each bidder i a payment equal to $\max(0, p_i^{PVM})$. Note, however, that lower-bounding the bidders' payments this way implies that the resulting mechanism is no longer (formally) social welfare-aligned. If the auctioneer chooses a sufficiently large number of queries, this violation of social welfare alignment is likely to be negligible for incentives in practice.

5. Instantiating the Machine Learning Algorithm

So far, our presentation of PVML has been agnostic regarding which particular ML algorithm to use in the elicitation module. However, this choice is very important because it determines not only the quality of the queries identified by Algorithm 1, but also whether running this algorithm is computationally practical. Indeed, each time the module Elicit$_{\mathcal{A}}$ is called, the mechanism needs to determine n social welfare-maximizing allocations in the virtual setting, each time under different constraints. In general, determining each of these allocations may require evaluating exponentially many allocations, and the evaluation of each allocation a requires applying $\mathcal{A}$ once per bidder i to derive each $\tilde{v}_i(a)$. However, if the learning model exhibits useful structure, we can exploit this structure when searching for the social welfare-maximizing allocations.

In this section, we present different learning models that can be integrated into our mechanism while keeping its computations practical. In Subsection 5.1, we start from the simple linear regression model. While computationally convenient, this learning model cannot capture complementarities between items, which prevents Algorithm 1 from inferring highly efficient allocations in settings where such complementarities are important. In Subsection 5.2 we introduce more *expressive* learning models, i.e., learning models able to capture more complex valuations. We show how our elicitation can be maintained computationally practical even under these new models.

5.1. Linear Regression

In linear regression, each inferred value function $\tilde{v}_i$ has the linear structure

$$\tilde{v}_i(x) = w_i \cdot x, \tag{8}$$

where $w_i \in \mathbb{R}^m$ is the *weight vector* corresponding to $\tilde{v}_i$.[9] Note that, given a weight vector w_i, each w_{ij} represents bidder i's inferred value for item j. A first, simple approach to determine $\tilde{v}_i(x)$ is to interpolate bidder i's reported values $\hat{v}_i = \{(x_{ik}, \hat{v}_{i,x_{ik}})\}_{k=1}^{\ell_i}$. We can then use a standard squared loss function $\mathcal{L}_2(y, \tilde{y}) = (y - \tilde{y})^2$ to quantify the interpolation error between observation y and prediction $\tilde{y}$, and choose our weight vector as

$$w_i \in \arg\min_{w_i'} \sum_{k=1}^{\ell_i} \mathcal{L}(\hat{v}_{ik}, w_i' \cdot x_{ik}). \tag{9}$$

While reasonable, this simple approach does not provide a full characterization of w_i and, consequently, of $\tilde{v}_i$. Indeed, the objective in (9) does not involve the weight w_{ij} of any item j that is not contained in any bundle x_{ik}. This means that w_i is randomly selected between different weight vectors that assign any value to item j. A standard approach to achieve determinism consists of convexifying the objective in (9) by adding an

[9] Here we do not introduce a bias term. This is consistent with assuming that our bidders always value zero the empty bundle.

extra term $||w_i||^2 = w_i \cdot w_i$. This approach is an instance of the Tikhonov regularization method (Tikhonov and Arsenin, 1977) and it biases the inference towards functions $\tilde{v}_i$ with low weights w_{ij}. The new learning model derived is called *regularized linear regression* and determines the weight vector as

$$w_i \in \arg\min_{w'_i} \; c \sum_{k=1}^{\ell_i} \mathcal{L}(\hat{v}_{ik}, w'_i \cdot x_{ik}) + ||w'_i||^2, \tag{10}$$

where $c > 0$ is a parameter of the model defining the trade-off between interpolation accuracy and regularization. Note that regularized linear regression avoids inferring unnecessary high values for items that are not contained in any bundle evaluated by bidder i.

When valuations are inferred via regularized linear regression, the optimization problem to determine $\tilde{a} \in \arg\max_{a \in \mathcal{F}} \sum_{i \in [n]} \tilde{v}_i(a)$ can be formulated as an integer programing problem (IP). The IP has $n \cdot m$ boolean variables a_{ij}, each denoting whether bidder i should get item j under allocation a, and m feasibility constraints enforcing that each item j is not allocated more than once:

$$\begin{aligned} \max_{a} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} a_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_{ij} \leq 1 \quad \forall j \in [m] \\ & a_{ij} \in \{0,1\} \quad \forall i \in [n], j \in [m]. \end{aligned} \tag{11}$$

To adapt IP (11) to the problem in Step 5 of Algorithm 1 it is sufficient to add an integer cut $\sum_{j \in [m]} |x_{ij} - a_{ij}| \geq 1$ for each bundle x that must not be allocated to bidder i in $\mathcal{F}_i$. While integer programming problems are NP-hard, using branch and bound algorithms (Land and Doig, 1960) as implemented by modern IP solving softwares such as CPLEX (CPLEX, 2019) allows us to solve Problem (11) in few milliseconds even for large auction instances with 98 items and 10 bidders.

Despite being computationally very convenient, linear regression models have a major drawback, i.e., they cannot capture valuations where two items are complements or substitutes.[10] In the next section, we see how these models can be generalized to effectively handle a broader class of valuations.

5.2. Support Vector Regression

Support vector regression (SVR) is a learning model that extends regularized linear regression to non-linear inferred valuations while allowing to maintain the computations in PVML practical. In this subsection we present the most important properties of SVR. We defer to Smola and Schölkopf (2004) for a detailed introduction.

To infer non-linear valuations, SVR algorithms project our bundle encodings in $\mathbb{R}^m$ into a high (possibly infinite) dimensional *feature space* $\mathbb{R}^f$. This projection is determined via a mapping function $\varphi : \mathbb{R}^m \to \mathbb{R}^f$ that is a parameter of the learning model. Each valuation function is then derived via linear regression in the feature space as

$$\tilde{v}_i(x) = w_i \cdot \varphi(x), \tag{12}$$

where $w_i \in \mathbb{R}^f$. The weight vector w_i is chosen such as to minimizes a potential similar to the one in (10), with the important difference that the squared loss is replaced by the ε-insensitive loss $\mathcal{L}_\varepsilon(y, \tilde{y}) = \max\{|y -$

[10] Under linear regression models, the inferred value bidder i has for bundle x is always additive as it is given by the sum of the inferred values w_{ij} for the items j contained in x (see Equation (8)).

$\tilde{y}|-\varepsilon, 0\}$, where $\varepsilon \geq 0$ is a parameter of the model.[11] As we will discuss below, the ε-insensitive loss allows to maintain the representation of the inferred valuation succinct. In standard machine learning applications, this is useful to speed up each evaluation $\tilde{v}_i(x)$. In our application, the ε-insensitive loss allows to maintain our mechanism computationally practical. We can now formulate the learning problem under SVRs as follows:

$$w_i \in \arg\min_{w_i'} \; c \sum_{k=1}^{\ell_i} \mathcal{L}_\varepsilon(\hat{v}_{ik}, w_i' \cdot \varphi(x_{ik})) + ||w_i'||^2. \tag{13}$$

When the dimensionality of the feature space is low, w_i can be determined via simplex methods, as for linear regression. However, SVR algorithms may involve high dimensional feature spaces, meaning that it is often impractical to determine $\tilde{v}_i$ via its weight vector. In this scenario, it is convenient to derive $\tilde{v}_i$ via the *dual version* of the learning problem presented in (13), which is formulated in ℓ_i pairs of variables α_{ik} and α^*_{ik} that respectively correspond to the two constraints $w_i' \cdot \varphi(x_{ik}) \leq \hat{v}_{ik} + \varepsilon$ and $w_i' \cdot \varphi(x_{ik}) \geq \hat{v}_{ik} - \varepsilon$:

$$\begin{aligned} \max_{\alpha_i, \alpha_i^*} \quad & -\frac{1}{2} \sum_{k,k'=1}^{\ell_i} (\alpha_{ik} - \alpha^*_{ik})(\alpha_{ik'} - \alpha^*_{ik'})\, \varphi(x_{ik}) \cdot \varphi(x_{ik'}) \\ & - \varepsilon \sum_{k=1}^{\ell_i} (\alpha_{ik} + \alpha^*_{ik}) + \sum_{k=1}^{\ell_i} \hat{v}_{i,x_{ik}} (\alpha_{ik} - \alpha^*_{ik}) \\ \text{s.t.} \quad & \alpha_{ik}, \alpha^*_{ik} \in [0, c] \quad \forall k \in [\ell_i]. \end{aligned} \tag{14}$$

Once the optimal α_i and α_i^* are determined, the inferred valuation can be formulated as

$$\tilde{v}_i(x) = \sum_{k=1}^{\ell_i} (\alpha_{ik} - \alpha^*_{ik}) \kappa(x, x_{ik}), \tag{15}$$

where the function $\kappa(x, x') = \varphi(x) \cdot \varphi(x')$ is commonly called *kernel function* and it implicitly computes the scalar product of x and x' in the feature space. When the kernel function can be evaluated via closed formulas, SVR algorithms can derive non-linear valuation functions $\tilde{v}_i$ while avoiding computations in high dimensional feature spaces. This means that φ is never directly derived, and we don't need to enumerate all the features to find the weights w_i.

The kernel function to employ in a specific application is usually determined by evaluating different options on the available data. As discussed by Williams and Rasmussen (2006, chapter 4), the kernel functions commonly considered belong to two classes:

- dot product kernel functions $\kappa(x, x') = \kappa(x \cdot x')$, and
- radial basis kernel functions (RBF kernels) $\kappa(x, x') = \kappa(||x - x'||)$.

In our application, the class of kernel functions to evaluate is selected based on two criteria: 1. the expressivity of the corresponding learning model (which is determined by the implicit feature mapping φ), and 2. the complexity of the optimization problem $\tilde{a} \in \arg\max_{a \in \mathcal{F}} \sum_{i \in [n]} \tilde{v}_i(a)$, which can be stated as:[12]

[11] Note that the ε-insensitive loss is null when the interpolation error is smaller than ε, thus inducing looser penalties.

[12] The instantiations of Problem (16) for dot product and radial basis kernel functions are provided in Appendix **??**.

$$\begin{aligned} \max_{a} \quad & \sum_{i=1}^{n}\sum_{k=1}^{\ell_i}(\alpha_{ik}-\alpha^*_{ik})\kappa(a_i,x_{ik}) \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_{ij} \leq 1 \quad \forall j \in [m] \\ & a_{ij} \in \{0,1\} \quad \forall i \in [n], j \in [m]. \end{aligned} \tag{16}$$

We identified three types of kernel functions. The first type is the one of *Gaussian kernel* functions. Gaussian kernels are commonly used in the machine learning literature:

Definition 5 (Gaussian Kernel)**.** *A Gaussian kernel is a kernel function of the form*

$$\kappa(x,x') = \exp\left(-\frac{||x-x'||^2}{\lambda}\right), \tag{17}$$

where λ is a non-negative parameter.

The Gaussian kernel is an RBF kernel with the important property that its induced learning model is *fully expressive*, i.e., it can capture *any* valuation function a bidder may have.

The second type of kernels we consider includes *Exponential kernels* which are also fully expressive but, unlike the Gaussian kernels, belongs to the family of dot-product kernel functions.

Definition 6 (Exponential Kernel)**.** *An exponential kernel is a kernel function of the form*

$$\kappa(x,x') = \exp\left(\frac{x\cdot x'}{\lambda}\right), \tag{18}$$

where λ is a non-negative parameter.

Last, we consider *Quadratic kernel* functions:

Definition 7 (Quadratic Kernel)**.** *A quadratic kernel is a kernel function of the form*

$$\kappa(x,x') = x\cdot x' + \lambda(x\cdot x')^2, \tag{19}$$

where λ is a non-negative parameter.

This kernel is not fully expressive, but it allows for the following convenient formulation of Problem (16):

$$\begin{aligned} \max_{a} \quad & \sum_{i=1}^{n}\sum_{k=1}^{\ell_i}(\bar{\alpha}_{ik}-\bar{\alpha}^*_{ik})\Big(a_i\cdot x_{ik} + \lambda(a_i\cdot x_{ik})^2\Big) \\ \text{s.t.} \quad & \sum_{i=1}^{n} a_{ij} \leq 1 \quad \forall j \in [m] \\ & a_{ij} \in \{0,1\} \quad \forall i \in [n], j \in [m]. \end{aligned} \tag{20}$$

Problem (20) is a quadratic programming problem with boolean variables. These problems can be formulated in analogous versions that can be linear or quadratic with concave objectives (Lima and Grossmann,

2017). Both linear and concave formulations are solvable via branch and bound methods. As discussed in Appendix **??**, the generic IP version of Problem (16) for dot-product kernels can be maintained arbitrarily succinct by employing a large ε in our SVR algorithms. However, as we will show experimentally in Section **??**, in large auction settings the ε needed to maintain Problem (16) computationally practical is usually too large to allow to properly infer bidders' valuations. Thus, the formulation presented in (20) is pretty appealing in large settings.

6. Experiments I: Optimizing the Machine Learning Algorithm

In order to evaluate PVML, we need to identify the kernel and parameters that work well for the machine learning algorithm. Before evaluating the full mechanism in the next section (7), we here perform a series of experiments to identify optimal settings.

6.1. Setup

In order to run these experiments we need data with which to exercise the mechanism. The allocation of spectrum is one of the most important applications of CAs. We therefore adopt the allocation of spectrum for our experimental evaluation. To do this, we employ the Spectrum Auction Test Suite (SATS) version 0.6.4 (Weiss, Lubin and Seuken, 2017), which allows us to easily generate thousands of auction instances on demand. SATS gives us access to each bidder's true values across all 2^m bundles, and it enables us to implicitly (without enumeration) compute an efficient allocation with respect to these *full* value profiles using succinct MIP formulations. This is important, as it enables to to report the efficiency of mechanisms with respect to a gold standard counter-factual calculation of the total social welfare obtainable in the domain. We tested our approach on three of SATS' value models across a range of complexity. We describe each in turn below:

- The Global Synergy Value Model (GSVM) (Goeree and Holt, 2008) generates medium-sized instances with 18 items and 7 bidders. GSVM models the items (spectrum licenses) as being arranged in two circles. Depending on her 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. GSVM also includes a "global" bidder with interest in two thirds of the total licenses.
- The Local Synergy Value Model (LSVM) (Scheffel, Ziegler and Bichler, 2012) also generates medium-sized instances with 18 items and 6 bidders. In LSVM the items are placed on a two-dimensional grid, and a bidder's value depends on a sigmoid function of the number of contiguous licenses near a target item of interest.
- The Multi-Region Value Model (MRVM) (Weiss, Lubin and Seuken, 2017) generates large instances with 98 items and 10 bidders. MRVM captures large settings, such as US and Canadian auctions, by modeling the licenses as being arranged in multiple regions and bands. Bidders' values are affected by both geography and frequency dimensions of the licenses. Depending on the type and on the number of licenses of interest, bidders are categorized as `national`, `regional` or `local`.

In order to implement our machine learning algorithm and mechanism allocation proceedures, we need to be be able to solve mixed integer programs (MIPs). In our experiments, we used CPLEX 12.9 (CPLEX, 2019) for this purpose. Experimentally we need to evalaute several different mechanism design choices, and do so with enough data points for statistical signficance. Consequently in our experiments we ran an enormous number of MIPs (approximately $2,000,000$). Accordingly, we adopt a modest timeout for the solver, which we set to 1 minute, and adopt the best solution found so far. We note that in practical use, auctioneers will typically have more time to let the optimizer run (typically at least an hour), which would improve the outcome of

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

Figure 2: Comparison of different kernels in the LSVM domain. The Quadratic kernel obtains the best efficiency under the imposed computational constraints.

our mechanisms; we thus report conservative results with respect to the optimality of the MIP solutions. We conducted our experiments on a computational grid with 20 machines. Each machine has two Intel E5-2680 v2 2.80GHz processors each of which has 10 physical and 20 logical cores.

6.2. Results

In the experiments in this section, we are interested in identifying which kernel to use in the ML algorithm such that the full mechanism will yield the most efficient outcome. In selecting a kernel, we need to optimize the various parameters of the learning algorithm (i.e. ε, C, and any kernel hyperparameters). We tune these parameters to maximize the effeciency of the predicted allocation. Because we need the machine learning algorithm to operate within a reasonable comptuational budget, one of the most important parameters is the insensitivity threshold ε. This is because, as is standard in kernel-based ML methods, ε controls the number of support vectors that are likely to be part of the learned model. The size of the winner determination MIP that we solve (see Appendix A) is heavily dependent on the number of support vectors. So ε determines a tradeoff between the learning error of the ML model, and the runtime of the winner determination optimizer. To investigate this tradeoff, we conducted an experiment on the two expensive kernels, Gaussian and Exponential, which we present in Figure 3, results for the other two domains are provided in Appendix B. To conduct this experiment, we first generate an LSVM domain using SATS. From this domain we then sample $Q \in \{100, 200, 500\}$ truthful bundle-value reports from each bidder. Based on these reports, we train an ML algorithm (with particular parameters) to construct $\tilde{v}$. We then evaluate this $\tilde{v}$ and its effectiveness in inferring optimal allocations (in terms of quality and speed) according to four measures of interest which we next describe.

First, we provide the efficiency of the inferred optimal allocation. To do this, we compute the social welfare of the gold standard optimal allocation at true values, $V(a_v^*)$, using the concise MIP formulation built into SATS, and compare this to the true value of the inferred optimal allocation, $V(a_{\tilde{v}^*})$. The table includes subcolumns for each Q, enabling us to compare the effect of training data size on the learning algorithm. For each kernel we show the results for three insensitivity thresholds, ε: the one that is best for efficiency, one half that size, and zero. From the table, we can see that most of the efficiency gains occur in the first half of the considered range of ε. Further, we see that for sufficiently large ε, efficiency is monotonically increasing in the sample size Q.

Next, we report the learning error of the machine learning algorithm. Here we are measuring learning error in the standard way by measuring the average absolute difference between the predicted and true value for all bundles in the domain.[13]. From the table we see that the higher the epsilon (and thus the fewer support vectors) the worse the learning error.

[13]This is possible for GSVM and LSVM which have 18 goods and are enumerable in this way; for MRVM we compute a similar statistic by sampling $100,000$ bundles.

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

Figure 3: The effect of varying the insensitivity parameter ε on the predictive performance: both efficiency of the predicted optimal allocation, and learning error. Results are shown for the two expensive kernels, Experimental and Gaussian, in the LSVM domain.[14]

Next, we report the solve time for the winner determination MIP that finds $a^*_{\tilde{v}}$ based upon the trained ML model. From the table we see that the solver always times out unless the insensitivity threshold ε is sufficiently large.

Finally, we list the optimality gap reported by the solver when it stops. Specifically, this is calculated as $(\overline{o} - \underline{o})/\underline{o}$, where $\overline{o}$, and $\underline{o}$ are the solver's proven upper and lower bounds on the optimal value respectively. When the value is zero, the solver has proven optimality. As expected the optimality gap is closely correlated with the solve time, but it provides a quantitative measure of the consequence of the solver stopping early.

Overall from the table, we see that it is important to select epsilon carefully in order to properly tradeoff learning performance (as measured via learning error) with solve time (as measured via WD Solve Time) in order to maximize the efficiency of the inferred optimal allocation (as measured by Efficiency).

Now that we have selected the best parameters for each kernel, we turn to a comparison of each of the kernels we have introduced for use with PVML. Accordingly, in each of our domains of study we have run PVML with Linear, Quadratic, Exponential and Gaussian kernels. We present the results for the LSVM domain in Figure 2; results for the other two domains are available in Appendix B. Following the result in the previous table, all perameters (i.e. ε, C, and any kernel hyperparameters) in this kernel comparison are tuned to maximize efficiency. We ran a similar experiment as in the previous table, by training with Q bundle-value pairs sampled uniformly at random from the domain, and then computing the same four outcome measures of interest.

From the table we can see the the Quadratic kernel yields the best efficiency at all sample sizes. The Linear kernel has the lowest efficiency, showing that more expensive kernels can be worth the computational effort they require.

Turning to the learning error column, we observe that all of entries in this table are higher than the lowest observed value in Table 3, which is the Exponential kernel with $\varepsilon = 0$. This indicates that the Exponential kernel is best able to generalize in this domain, but its calculation is sufficiently expensive that we are generally better off choosing a Quadratic kernel that has somewhat higher learning error, but a more succient winner determination formulation.

We note that efficiency is typically inversely related to learning error. For example, the efficiency of the Quadratic kernel rises in the sample size Q, while its learning error decreases in Q. We note that this isn't always the case, for example in the Linear kernel, where increasing sample size yields lower learning error, but where the imputed optimal allocation actually becomes less efficient.

[14]Dashes in the table indicate entries where we could not get complete results because CPLEX could not find any feasible solution within the given time limit.

Turning to the final columns in the table, we see that the Linear kernel is solvable very rapdly with 0 optimality gap. Whereas the Exponential and Gaussian kernels are more expensive.

Overall, the Quadratic kernel has the best tradeoff between learning error and solve time, yielding the most efficient imputed optimality allocation within our computational budget, and we therefore adopt it going forward in the experients in the next section.

7. Experiments II: Comparing PVML against CCA

In this section, we turn to evaluating the full PVML mechanism in comparison to the widely used CCA.

7.1. Setup

To run our experiments we need to specify a number of attributes of both the PVML and CCA mechanisms, as well as define the scope and properties of the experiements themselves.

PVML. PVML is parametrized by the machine learning algorithm $\mathcal{A}$, the number of queries $Q_{\max}$ and the number of initial queries Q_0. As we described in Section 6, our ML algorithm will be an SVR with Quadratic kernel for all domains of study. In our experiments we will consider values for $Q_{\max} \in \{100, 200, 500\}$.[15] We note that the $Q_{\max} = 500$ is a practical number of queries, given that it was used in the real-world Canadian auction that inspired the MRVM model that we use. We include the smaller values for comparison. For the number of initial queries Q_0, we select an optimal value through offline tuning. Specifically, we use $Q_0 = 90$ for GSVM, $Q_0 = 60$ for LSVM, and $Q_0 = 50$ for MRVM.

CCA. The CCA is parametrized by the reserve prices employed, the way prices are updated, and what heuristics are assumed for bidder behavior in the supplementary phase. In our experiments, we use reserve prices for each license equal to 1% of the average license value derived from $10,000$ bundle-value pairs sampled from bidders in the domain. For price updates in our CCA, we follow the parametrization of the real-world Canadian CCA, that used 5% price increments. We note that the number of auction rounds we obtain under these price settings is similar to the number of auction rounds observed in the real-world Canadian auction. Finally, we specify how bidders in the CCA select which bundle-value pairs to report in the supplementary round.[16] We consider three different *bundle selection heuristics*:

- ClockBids: Include in the supplemental round only those bundles reported in the clock phase (as an answer to the corresponding demand query), at a value equal to the highest quoted price for that bundle.
- ClockBidsRaised: bidders provide their true values for the unique bundles reported during the clock phase.
- ProfitMax: bidders report their true values for the Q_{PM} bundles earning them the highest profit at the final clock prices. To get a similar number of reported values across small and large bidders, we also let bidders report values for bundles earning them negative profit. In our simulations we use $Q_{PM} = \{100, 200, 500\}$.

[15] We note that we parameterized the experiments in the previous section by a training set size Q, that serves a similar role to $Q_{\max}$, but is a simpler concept.

[16] There is no prior work to guide the optimal strategy for bidders in choosing bundles to bid upon in the supplemental round. We therefore explored several different heuristics in our experiments after consulting with industry experts who have been involved in the design of the CCA and who have provided advice to bidders in the CCA. The lack of theoretical guidance here represents an additional strategic burden on bidders, in contrast to PVM.

Experimental Framework. Throughout this section, we simulate straightforward, truthful bidding both in PVML and the CCA. To be conservative, we don't allow any bidders to "push" bundles in our PVML evalaution. This avoids confusing the results by including contributions due to a bidder push heuristic.

Our experiments are run on the same computational grid as was used in Section 6. The experiments in this section required a very significant computational effort, with $162,000$ core hours employed.

7.2. Results

We now consider each of our three domains in turn. We start with the LSVM domain, for which we present results in Table 4. To run the experiment we sample 100 instances drawn randomly from the domain. We then compute the outcome for these instances in each mechanism, and report the mean value on each row. We note that this results in a paired sampling process which improves the statistical signficance of comparisons between the data points.

Mechanism	Heuristic	Query Cap	Efficiency	Revenue	Revenue (Core)	Rounds
VCG			100.0%	83.2%	-	1
CCA	Clock Bids		81.4%	59.9%	64.7%	124
	Clock Bids Raised		90.9%	74.6%	79.0%	124
	Profit Max	100	99.4%	78.8%	82.9%	124
	Profit Max	200	99.8%	81.0%	85.0%	124
	Profit Max	500	99.9%	82.2%	86.3%	124
PVML		100	98.6%	69.4%	-	13
		200	99.1%	76.2%	-	37
		500	99.7%	76.9%	-	113

Figure 4: Comparison between the VCG, CCA and the PVML mechanism in the LSVM domain.

From the table, we can see the CCA obtains almost perfect efficiency, but that with sufficiently many samples, PVML is nearly as good. We note that the strong performance of the CCA is heavily dependent on the employed Profit Max heuristic for the supplementary phase. In the table, we next report the fraction of surplus acruing to the seller. For the CCA, in this column we use a VCG-style payment rule on the reported bundle-value pairs from the supplementary phase, which produces a payment profile most similar to VCG itself (first row), and the payments obtained in PVML (bottom rows). In LSVM, VCG produces a high level of revenue, and we see that the other mechanisms approximately match it (apart from the ClockBids heuristic of the CCA). In the next column we show a similar revenue number for the VCG-nearest core-selecting rule as applied to the bundle-value pairs in the supplemental round of the CCA (this is the payment rule that is often used in practice). As expected, we see some revenue lift when swapping VCG for the core-selecting rule. Finally the table shows the number of rounds employed by each mechanism. We see that PVML uses fewer rounds than the CCA, illustrating that it is a practical mechanism.

Overall, in the LSVM domain we observe that the CCA does well. This is likely due to bidders in the LSVM domain having strongly peaked preferences around a particular subset of bundles, which bidders can describe even when only stating 100 bundle-value pairs, as they can even in our setting with the smallest query cap.

Next we turn to the GSVM domain, the results for which we show in Table 5.

Mechanism	Heuristic	Query Cap	Efficiency	Revenue	Revenue (Core)	Rounds
VCG			100.0%	82.4%	-	1
CCA	Clock Bids		94.2%	60.5%	67.6%	118
	Clock Bids Raised		96.8%	78.2%	82.8%	118
	Profit Max	100	99.2%	78.4%	82.6%	118
	Profit Max	200	99.6%	79.6%	83.3%	118
	Profit Max	500	99.7%	81.2%	84.2%	118
PVML		100	100.0%	81.2%	-	6
		200	100.0%	82.4%	-	41
		500	100.0%	82.4%	-	153

Figure 5: Comparison between the VCG, CCA and the PVML mechanism in the GSVM domain.

In the table, we can see that PVML provides 100% efficiency even with only a 100 query cap. The CCA with the Profit Max heuristic is also performing well. Turning to revenue, we once again see that VCG produces a high level of revenue in this setting, and that all of the CCA and PVML versions are close to VCG, except for the ClockBids heuristic (as we saw in LSVM). Again, and as expected, we observe that the core-selecting payment rule produces a revenue lift. Finally, we observe that PVML requires a similar number of rounds as the CCA, although slightly more with a 500 query cap.

Importantly, we note that bidder preferences in the GSVM domain can be fully captured by the quadratic kernel (see the proposition in the appendix of Brero, Lubin and Seuken (2017)). Whenever preferences can be fully captured in this way, with a high enough query cap, PVML efficiency will reach 100%; GSVM is simple enough that this is already reached with only 100 queries.

Finally, we turn to the larger, more realistic, and more complex MRVM domain, the results for which we show in Figure 6.

Mechanism	Heuristic	Query Cap	Efficiency	Revenue	Revenue (Core)	Rounds
VCG			100.0%	42.4%	-	1
CCA	Clock Bids		93.0%	17.3%	17.9%	140
	Clock Bids Raised		93.2%	30.3%	35.4%	140
	Profit Max	100	92.0%	5.5%	5.7%	140
	Profit Max	200	92.1%	6.1%	6.2%	140
	Profit Max	500	92.4%	6.9%	7.0%	140
PVML		100	91.5%	20.9%	-	13
		200	93.3%	26.2%	-	25
		500	94.6%	31.6%	-	56

Figure 6: Comparison between the VCG, CCA and the PVML mechanism in the MRVM domain.

MRVM has 2^{98} bundles, and, importantly, these bundles are *not* strongly peaked. This means 500 bundle-value pairs is not sufficient to reasonably describe bidder preferences. As a consequence the CCA Profit Max heuristic is not nearly as effective as it was in the simper domains. Moreover, Profit Max is not the best heuristic for the CCA, as it previously was. We note that this means that a bidder cannot adopt a best heuristic apriori, but must instead taylor her strategy to the domain they are operating in (unlike in PVML). Even still, PVML is beating the best CCA heuristic on effiency by 1.4%. Turning to revenue, we observe that the CCA with the Profit Max heuristic obtains only a very small revenue. Further, we observe that this low

level of revenue is present not only with the VCG payment rule, but also with a core-selecting payment rule. By contrast, VCG yields a reasonable amount of revenue in this setting, a level nearly matched by PVML. Finally, we observe that even in this very complex domain, PVML only requires a very modest number of rounds, significantly fewer than the number required by the CCA.

Overall we see that the PVML mechanism performs exceptionally well in this complex domain. We see that when the domain is very large and complex, the rich ML-based elicitation can be highly effective, while linear pricing, as used by the CCA, becomes less effective.

8. Conclusion

In this paper, we have introduced a machine learning-powered iterative combinatorial auction mechanism. In contrast to prior designs like the combinatorial clock auction (CCA), our auction does not use prices but *value queries* to interact with the bidders. Via simulations, we have shown that our ML-powered auction is able to achieve higher efficiency than the CCA, even with just a small number of queries per bidder.

Two components in our design are responsible for this efficiency gain: (1) the ML algorithm infers a bidder's value function on the whole bundle space, and (2) in each iteration of the auction, we compute the (tentatively) optimal allocation based on the inferred values, which we use to decide which value query to ask next to each bidder. To achieve good incentives, we have drawn on principles from the VCG mechanism to design *PVM*, which aligns bidders' incentives with allocative efficiency and is also very robust against strategic manipulations in practice.

Our results give rise to promising directions for future research: First, while we have shown how our elicitation method can be modified to allow bidders to only report bounds on their values, we have left a full mechanism based on this bounds-based elicitation method to future work. Second, while our method of selecting the *initial set of queries* uniformly at random from the whole bundle space has worked surprisingly well, future work could explore more sophisticated active learning methods for generating this initial set.

The design of a full auction mechanism based on upper-and-lower bounds queries is one of the future research directions based on this paper.

Appendices

A. Winner Determination for Dot-Product and RBF Kernels

In this appendix we show how to formulate Problem (16) under dot-product and RBF kernels as integer programming problems (IPs). 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 consider and our binary bundle encoding.

We start by noticing that, under dot-product kernel functions, each kernel evaluation in the objective of Problem (16) can be formulated as $\kappa(x, x_{ik}) = \kappa(x \cdot x_{ik}) = \kappa(\tau)$, where $\tau \in \{0, ..., \bar{\tau}\}$ is the number of items bundles x and x_{ik} have in common, and $\bar{\tau}$ is the size of bundle x_{ik}. Similarly, under RBF kernels, we have that each kernel evaluation in Problem (16) can be formulated as $\kappa(x, x_{ik}) = \kappa(||x - x_{ik}||) = \kappa(\tau)$, where $\tau \in \{0, ..., \bar{\tau}\}$ is the number of items contained in only one among bundles x and x_{ik}, and $\bar{\tau}$ is the total number of items m. In both cases, we can introduce $\bar{\tau} + 1$ binary variables $z_{ik\tau}$, each indicating the value τ where the kernel function should be evaluated, and encode each kernel evaluation $\kappa(x, x_{ik})$ as

$$\begin{aligned} & \sum_{\tau=0}^{\bar{\tau}} \kappa(\tau) z_{ik\tau} \\ \text{s.t.} \quad & \sum_{\tau=0}^{\bar{\tau}} z_{ik\tau} = 1. \end{aligned} \tag{21}$$

To use this linearized kernel encoding in Problem (16), we should also establish the relationship between the allocation variables a_{ij} and the newly introduced binary variables $z_{ik\tau}$. Under dot-product kernels, this relationship can be encoded by introducing the following constraint for each support vector x_{ik}:

$$\sum_{j \in x_{ik}} a_{ij} = \sum_{\tau=0}^{|x_{ik}|} (\tau + 1) z_{ik\tau} - 1. \tag{22}$$

The left term in equation (22) tracks the number of items that bidder i's allocated bundle a_i and the support vector x_{ik} have in common; the right term enforces that only the $z_{ik\tau}$ corresponding to this number gets activated. Under RBF kernels, the relationship between variables a_{ij} and $z_{ik\tau}$ for each x_{ik} can be encoded as

$$\sum_{j \in x_{ik}} (1 - a_{ij}) + \sum_{j \notin x_{ik}} a_{ij} = \sum_{\tau=0}^{m} (\tau + 1) z_{ik\tau} - 1. \tag{23}$$

Here, the left term tracks the number of items that belong to x_{ik} and not to a_{ij} (first sum) and the ones that belong to a_{ij} and not to x_{ik} (second sum).

Integrating the kernel encoding in (21) and the constraints in (22) in Problem (16), we have that the alloca-

tion problem for dot-product kernels can be encoded via the following IP:

$$\max_{a_{ij}, z_{ik\tau}} \quad \sum_{i=1}^{n} \sum_{k=1}^{\ell_i} (\alpha_{ik} - \alpha^*_{ik}) \sum_{\tau=0}^{\bar{\tau}} \kappa(\tau) z_{ik\tau} \qquad \text{Problem (16)}$$

$$\text{s.t.} \quad \sum_{\tau=0}^{\bar{\tau}} z_{ik\tau} = 1 \quad \forall i \in [n],\, k \in [\ell_i] \qquad \text{Encoding (21)}$$

$$\sum_{j \in x_{ik}} a_{ij} = \sum_{\tau=0}^{|x_{ik}|} (\tau + 1) z_{ik\tau} - 1 \quad \forall i \in [n],\, k \in [\ell_i] \qquad \text{Equation (22)}$$

$$\sum_{i=1}^{n} a_{ij} \leq 1 \quad \forall j \in [m] \qquad \text{Problem (16)}$$

By replacing the constraints in (22) with the constraints in (23), we obtain the following formulation of the allocation problem under RBF kernels:

$$\max_{a_{ij}, z_{ik\tau}} \quad \sum_{i=1}^{n} \sum_{k=1}^{\ell_i} (\alpha_{ik} - \alpha^*_{ik}) \sum_{\tau=0}^{\bar{\tau}} \kappa(\tau) z_{ik\tau} \qquad \text{Problem (16)}$$

$$\text{s.t.} \quad \sum_{\tau=0}^{\bar{\tau}} z_{ik\tau} = 1 \quad \forall i \in [n],\, k \in [\ell_i] \qquad \text{Encoding (21)}$$

$$\sum_{j \in x_{ik}} (1 - a_{ij}) + \sum_{j \notin x_{ik}} a_{ij} = \sum_{\tau=0}^{m} (\tau + 1) z_{ik\tau} - 1 \quad \forall i \in [n],\, k \in [\ell_i] \qquad \text{Equation (23)}$$

$$\sum_{i=1}^{n} a_{ij} \leq 1 \quad \forall j \in [m] \qquad \text{Problem (16)}$$

Note that the size of both the integer programming problems presented above heavily depends on the number of support vectors. In both problems, each support vector x_{ik} introduces two constraints and $|x_{ik}| + 1$ binary variables under dot-product kernels or $m + 1$ binary variables under RBF kernels. As discussed in Section 5, one can reduce the number of support vectors by using a larger ε in the SVR training problem (16) which can be extremely helpful to maintain our mechanism computationally tractable.

B. Machine Learning Algorithm Comparison: Additional Results

B.1. GSVM

Kernel	Efficiency			Learning Error			WD Solve Time			Optimality Gap		
	100	200	500	100	200	500	100	200	500	100	200	500
Linear	83.8%	83.9%	84.0%	14.08	13.09	12.57	0.00s	0.00s	0.01s	0.00	0.00	0.00
Quadratic	99.9%	100.0%	100.0%	1.60	0.00	0.02	0.09s	0.06s	0.07s	0.00	0.00	0.00
Exponential	97.1%	97.3%	97.8%	12.09	10.46	9.73	7.15s	12.08s	19.02s	0.00	0.00	0.00
Gaussian	88.3%	89.0%	89.7%	12.72	10.81	9.78	2.41s	3.25s	7.72s	0.00	0.00	0.00

Figure 7: GSVM

Kernel	ϵ	Efficiency			Learning Error			WD Solve Time			Optimality Gap		
		100	200	500	100	200	500	100	200	500	100	200	500
Exponential	0	94.0%	91.8%	81.5%	2.24	1.03	0.49	59.42s	60.00s	60.00s	0.31	1.29	7.02
Exponential	8	96.3%	96.7%	97.3%	7.46	5.97	5.24	12.32s	16.00s	26.06s	0.00	0.00	0.01
Exponential	16	88.3%	89.0%	89.7%	12.09	10.46	9.73	7.15s	12.08s	19.02s	0.00	0.00	0.00
Gaussian	0	84.7%	85.3%	78.3%	4.77	2.16	0.38	60.00s	60.00s	60.00s	0.96	2.81	5.56
Gaussian	8	88.2%	88.1%	88.9%	8.18	6.31	5.20	10.69s	29.07s	51.21s	0.01	0.04	0.15
Gaussian	16	88.3%	89.0%	89.7%	12.72	10.81	9.78	2.41s	3.25s	7.72s	0.00	0.00	0.00

Figure 8: GSVM

B.2. MRVM

Kernel	Efficiency			Learning Error			WD Solve Time			Optimality Gap		
	100	200	500	100	200	500	100	200	500	100	200	500
Linear	71.6%	73.2%	69.8%	1855.25	800.01	608.88	0.01s	0.01s	0.01s	0.00	0.00	0.00
Quadratic	72.6%	79.2%	82.2%	1000.12	525.27	339.27	10.72s	60.00s	60.00s	0.00	0.16	0.39
Exponential	66.7%	65.9%	64.0%	1987.05	1913.78	1831.73	35.09s	53.60s	60.00s	0.00	0.01	0.04
Gaussian	78.9%	79.4%	78.0%	2089.60	1983.68	1871.82	60.00s	60.00s	60.00s	0.11	0.17	0.28

Figure 9: MRVM

Kernel	ϵ	Efficiency			Learning Error			WD Solve Time			Optimality Gap		
		100	200	500	100	200	500	100	200	500	100	200	500
Exponential	0	62.8%	51.1%	3.9%	593.91	406.28	295.33	60.00s	60.00s	60.00s	0.21	0.74	3.17
Exponential	2048	65.4%	63.9%	61.8%	1462.45	1348.47	1226.41	60.00s	60.00s	60.00s	0.04	0.08	0.00
Exponential	4096	66.7%	65.9%	64.0%	1987.05	1913.78	1831.73	35.09s	53.60s	60.00s	0.00	0.00	0.04
Gaussian	0	-	-	-	-	-	-	-	-	-	-	-	-
Gaussian	2048	79.7%	77.7%	-	1568.93	1423.69	-	60.00s	60.00s	-	0.22	0.36	-
Gaussian	4096	78.9%	79.4%	78.0%	2089.60	1983.68	1871.82	60.00s	60.00s	60.00s	0.11	0.17	0.28

Figure 10: MRVM

References

CPLEX. 2019. *https://www.ibm.com/analytics/cplex-optimizer*, Retrieved 5/11/19.

Ausubel, Lawrence M. 2006. "An efficient dynamic auction for heterogeneous commodities." *American Economic Review*, 96(3): 602–629.

Ausubel, Lawrence M, and Oleg Baranov. 2017. "A practical guide to the combinatorial clock auction." *The Economic Journal*, 127(605).

Ausubel, Lawrence M, Peter Cramton, Paul Milgrom, et al. 2006. "The Clock-proxy Auction: A Practical Combinatorial Auction Design." In *Combinatorial Auctions*, ed. Peter Cramton, Yoav Shoham and Richard Steinberg, Chapter 5. MIT Press.

Balcan, Maria-Florina F, Tuomas Sandholm, and Ellen Vitercik. 2016. "Sample complexity of automated mechanism design." In *Advances in Neural Information Processing Systems*.

Balcan, Maria-Florina, Tuomas Sandholm, and Ellen Vitercik. 2017. "A General Theory of Sample Complexity for Multi-Item Profit Maximization." *arXiv preprint arXiv:1705.00243*.

Bichler, Martin, Pasha Shabalin, and Jürgen Wolf. 2013. "Do Core-selecting Combinatorial Clock Auctions Always Lead to High Efficiency? An Experimental Analysis of Spectrum Auction Designs." *Experimental Economics*, 16(4): 511–545.

Blum, Avrim, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. 2004. "Preference elicitation and query learning." *Journal of Machine Learning Research*, 5: 649–667.

Brero, Gianluca, and Sébastien Lahaie. 2018. "A Bayesian Clearing Mechanism for Combinatorial Auctions." In *Proceedings of the Thirty-second AAAI Conference of Artificial Intelligence*.

Brero, Gianluca, Benjamin Lubin, and Sven Seuken. 2017. "Probably Approximately Efficient Combinatorial Auctions via Machine Learning." In *Proceedings of the Thirty-first AAAI Conference of Artificial Intelligence*.

Brero, Gianluca, Benjamin Lubin, and Sven Seuken. 2018. "Combinatorial Auctions via Machine Learning-based Preference Elicitation." In *Proceedings of the Twenty-seventh International Joint Conference on Artificial Intelligence*.

Brero, Gianluca, Sébastien Lahaie, and Sven Seuken. 2019. "Fast Iterative Combinatorial Auctions via Bayesian Learning." In *Proceedings of the Thirty-third AAAI Conference on Artificial Intelligence*.

Cesa-Bianchi, Nicolo, Claudio Gentile, and Yishay Mansour. 2015. "Regret minimization for reserve prices in second-price auctions." *IEEE Transactions on Information Theory*, 61(1): 549–564.

Clarke, Edward H. 1971. "Multipart pricing of public goods." *Public choice*, 11(1): 17–33.

Cole, Richard, and Tim Roughgarden. 2014. "The sample complexity of revenue maximization." In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*.

Conitzer, Vincent, and Tuomas Sandholm. 2002. "Complexity of mechanism design." In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*. Edmonton, Canada.

Conitzer, Vincent, and Tuomas Sandholm. 2004. "Self-interested automated mechanism design and implications for optimal combinatorial auctions." In *Proceedings of the 5th ACM conference on Electronic commerce*.

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

de Vries, Sven, James Schummer, and Rakesh V Vohra. 2007. "On ascending Vickrey auctions for heterogeneous objects." *Journal of Economic Theory*, 132(1): 95–118.

Duetting, Paul, Zhe Feng, Harikrishna Narasimhan, and David C. Parkes. 2018. "Optimal Auctions through Deep Learning." arXiv:1706.0345.

Dütting, Paul, Felix Fischer, Pichayut Jirapinyo, John K Lai, Benjamin Lubin, and David C Parkes. 2015. "Payment rules through discriminant-based classifiers." *ACM Transactions on Economics and Computation*, 3(1): 5.

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

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

Golowich, Noah, Harikrishna Narasimhan, and David C Parkes. 2018. "Deep Learning for Multi-Facility Location Mechanism Design." In *IJCAI*.

Groves, Theodore. 1973. "Incentives in teams." *Econometrica: Journal of the Econometric Society*, 617–631.

Industry Canada. 2013. "Responses to Clarification Questions on the Licensing Framework for Mobile Broadband Services (MBS) — 700 MHz Band." Government of Canada.

Janssen, Maarten, and Bernhard Kasberger. 2015. "On the Clock of the Combinatorial Clock Auction." Working Paper.

Janssen, Maarten, Vladimir Karamychev, and Bernhard Kasberger. 2017. "Handbook of Spectrum Auction Design." , ed. M. Bichler and J. Goeree, Chapter Budget Constraints in Combinatorial Clock Auctions, 320–337. Cambridge University Press.

Knapek, Stephan, and Achim Wambach. 2012. "Strategic Complexities in the Combinatorial Clock Auction." CESifo Working Paper 3983.

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

Lahaie, Sébastien, and Benjamin Lubin. 2019. "Adaptive-price combinatorial auctions." In *Proceedings of the 2019 ACM Conference on Economics and Computation*.

Lahaie, Sébastien, and David C Parkes. 2004. "Applying Learning Algorithms to Preference Elicitation." In *Proceedings of the 5th ACM Conference on Electronic Commerce*.

Land, Ailsa H, and Alison G Doig. 1960. "An automatic method of solving discrete programming problems." *Econometrica: Journal of the Econometric Society*, 497–520.

Lima, Ricardo M, and Ignacio E Grossmann. 2017. "On the solution of nonconvex cardinality Boolean quadratic programming problems: a computational study." *Computational Optimization and Applications*, 66(1): 1–37.

Mas-Colell, Andreu, Michael D. Whinston, and Jerry R. Green. 1995. *Microeconomic theory.* Oxford university Press.

Medina, Andres M, and Mehryar Mohri. 2014. "Learning theory and algorithms for revenue optimization in second price auctions with reserve." In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*.

Mishra, Debasis, and David C Parkes. 2007. "Ascending price Vickrey auctions for general valuations." *Journal of Economic Theory*, 132(1): 335–366.

Morgenstern, Jamie, and Tim Roughgarden. 2016. "Learning simple auctions." In *Conference on Learning Theory*.

Morgenstern, Jamie H, and Tim Roughgarden. 2015. "On the pseudo-dimension of nearly optimal auctions." In *Advances in Neural Information Processing Systems*.

Narasimhan, Harikrishna, Shivani Brinda Agarwal, and David C Parkes. 2016. "Automated mechanism design without money via machine learning." In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

Nisan, Noam. 2006. "Bidding languages." *Combinatorial auctions. Cambridge: MIT*.

Nisan, Noam, and Amir Ronen. 2007. "Computationally feasible VCG mechanisms." *Journal of Artificial Intelligence Research*, 29: 19–47.

Nisan, Noam, and Ilya Segal. 2006. "The communication requirements of efficient allocations and supporting prices." *Journal of Economic Theory*, 129(1): 192–224.

Parkes, David C. 2001. "An iterative generalized Vickrey auction: Strategy-proofness without complete revelation." In *Proceedings of AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents*.

Parkes, David C. 2006. "Iterative Combinatorial Auctions." In *Combinatorial Auctions*, ed. Peter Cramton, Yoav Shoham and Richard Steinberg, Chapter 2. MIT Press.

Roughgarden, Tim, and Okke Schrijvers. 2016. "Ironing in the Dark." In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*.

Sandholm, T. 2013. "Very-Large-Scale Generalized Combinatorial Multi-Attribute Auctions: Lessons from Conducting $60 Billion of Sourcing." In *The Handbook of Market Design*, ed. Nir Vulkan, Alvin E. Roth and Zvika Neeman, Chapter 1. Oxford University Press.

Sandholm, Tuomas, and Craig Boutilier. 2006. "Preference Elicitation in Combinatorial Auctions." In *Combinatorial Auctions*, ed. Peter Cramton, Yoav Shoham and Richard Steinberg, Chapter 10. The MIT Press.

Scheffel, Tobias, Georg Ziegler, and Martin Bichler. 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.

Smola, Alex J, and Bernhard Schölkopf. 2004. "A tutorial on support vector regression." *Statistics and computing*, 14(3): 199–222.

Syrgkanis, Vasilis. 2017. "A sample complexity measure with applications to learning optimal auctions." In *Advances in Neural Information Processing Systems*.

Tikhonov, Andrey N, and Vasilii Iakkovlevich Arsenin. 1977. *Solutions of ill-posed problems.* Vol. 14, Vh Winston.

Vickrey, William. 1961. "Counterspeculation, auctions, and competitive sealed tenders." *The Journal of finance*, 16(1): 8–37.

Weiss, Michael, Benjamin Lubin, and Sven Seuken. 2017. "SATS: A Universal Spectrum Auction Test Suite." In *Proceedings of the Sixteenth Conference on Autonomous Agents and MultiAgent Systems*.

Williams, Christopher K., and Carl E. Rasmussen. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.