Journal of Machine Learning Research 5 (2004) 649-667 Submitted 1/04; Published 6/04

Preference Elicitation and Query Learning

Avrim Blum AVRIM@CS.CMU.EDU
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA

Jeffrey Jackson JACKSON@MATHCS.DUQ.EDU
Department of Mathematics and Computer Science
Duquesne University
Pittsburgh, PA 15282-0001, USA

Tuomas Sandholm SANDHOLM@CS.CMU.EDU
Martin Zinkevich MAZ@CS.CMU.EDU
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891, USA

Editor: Kristin Bennett and Nicolò Cesa-Bianchi

Abstract

In this paper we explore the relationship between "preference elicitation", a learning-style problem that arises in combinatorial auctions, and the problem of learning via queries studied in computational learning theory. Preference elicitation is the process of asking questions about the preferences of bidders so as to best divide some set of goods. As a learning problem, it can be thought of as a setting in which there are multiple target concepts that can each be queried separately, but where the goal is not so much to learn each concept as it is to produce an "optimal example". In this work, we prove a number of similarities and differences between two-bidder preference elicitation and query learning, giving both separation results and proving some connections between these problems.

Keywords: exact learning, query learning, combinatorial auctions, preference elicitation.

1. Introduction

In a combinatorial auction, an entity (the "auctioneer") has a set S of n items that he would like to partition among a set of k bidders. What makes an auction *combinatorial* is that the valuations of the bidders — how much they would be willing to pay for different subsets of items — may not necessarily be linear functions over the items. For instance, if item a is a left shoe and item b is a right shoe, then a bidder might be willing to pay a reasonable amount for the bundle $\{a, b\}$ but very little for just $\{a\}$ or just $\{b\}$. In the other direction, if a and b are each *pairs* of shoes, then a bidder might value $\{a, b\}$ less than the sum of his valuations on $\{a\}$ and $\{b\}$ (especially if he just needs one pair of shoes right now). A standard goal for the auctioneer in such a setting is to determine the allocation of goods that maximizes *social welfare*: this is the sum, over all bidders, of the value that each bidder places on the set of items that he receives. This goal is perhaps most natural if one thinks of the auctioneer as not having a financial interest of its own but simply as an agent acting to help divide up a given set of items in a way that maximizes overall happiness or value. For example, the auctioneer might be a government agency, or a manager dividing up resources among different projects. The case of $k = 2$ can also be thought of as a situation in which one of the bidders represents a buyer (with various preferences over bundles of items) and the other bidder represents a marketplace (with various discounts and package-deals), and the auctioneer is helping the buyer decide what subset of items to purchase from the marketplace.[1]

There are a number of issues that arise in the combinatorial auction setting. For example, there is much work on designing protocols (mechanisms) so that bidders will be truthful in reporting their valuations and not want to "game" the system (Sandholm, 2002b, Nisan and Ronen, 2000, Lehmann et al., 2002). But another issue is that even if we can get bidders to be truthful, their valuation functions can be quite complicated. Because of this, bidding in a traditional manner may require an exponential amount of communication. This has led researchers to study the notion of preference elicitation, in which the auctioneer asks questions of the bidders in order to learn (elicit) enough information about their preferences to determine the best, or approximately best, allocation of the items (Conen and Sandholm, 2001). Because the issues of truthfulness can be handled in this setting via known mechanisms (discussed later in Section 6), the problem then becomes one of determining which questions to ask in order to extract the information needed for allocation, and to understand when this can be done quickly.

1.1 Preference Elicitation and Query Learning

We can think of preference elicitation in the context of query learning by thinking of each item as a boolean feature, thinking of a subset of the n items (a "bundle") as an example $x \in \{0,1\}^n$ indicating which items are in the subset, and thinking of the bidder's valuation function as a target function. The standard assumption of "free disposal" — bidders can throw away items for free — means we can assume that these valuation functions are *monotone*, though they typically will not be boolean-valued (we might have $f(x) = \$100$, for instance). Furthermore, one of the natural types of queries studied in preference elicitation, the *value query* (where the auctioneer asks the bidder how much he values some bundle), corresponds exactly with the learning-theoretic notion of a membership query.

On the other hand, a key difference between preference elicitation and query learning is in the goals. In learning, the objective is typically to recover the target function, either exactly or approximately. In preference elicitation, however, the goal is more one of finding the "best example". For instance, if there are just two bidders with preference functions f and g, then the goal is to find a partition (S', S'') of the n items to maximize $f(S') + g(S'')$. Thinking in terms of functions over $\{0,1\}^n$, the goal is to find $x \in \{0,1\}^n$ to maximize $f(x) + g(\bar{x})$.

Notice that one of the immediate differences between preference elicitation and query learning is that preference elicitation makes sense even if the target functions do not have short descriptions, or even short approximations. We will see some interesting examples

1. To think of this as a combinatorial auction, it is easiest to imagine that the auctioneer has pre-purchased *all* the items, and then is deciding which the buyer should keep and which should be returned for a refund.

later, but as a simple case, if we learn that bidder A will pay \$100 for the entire set of n items but no more than \$50 for any subset of size $n-1$ (she is a collector and wants the whole set), and B will pay a maximum of \$50 even for the whole lot, then we know we might as well give all items to A, and we do not need to know exactly how much each bidder would have paid for different subsets. On the other hand, it is quite possible for allocation of items to be *computationally* hard, even if the preferences of all the bidders are known. For example, even if each bidder's preferences can be expressed as a simple conjunction (these are called "single-minded" bidders), then if there are many bidders, allocation is equivalent to the NP-hard set-packing problem. Even for two bidders, allocation can be NP-hard for somewhat more complicated preference functions, such as read-once formulas (Zinkevich et al., 2003).

Another difference concerns the types of queries that are most natural in each setting. While value/membership queries are common to both, equivalence queries are quite *un*natural in the context of preference elicitation. On the other hand, the *demand query*, a powerful type of query for preference elicitation introduced by Nisan and Segal (2003), does not seem to have been studied in query learning.[2]

In this paper, we discuss similarities and differences between the three objectives of exact learning, approximate learning, and preference elicitation. We then give a number of upper and lower bounds for preference elicitation of natural preference (concept) classes. We focus primarily on the case of $k = 2$ bidders, because even this case is quite interesting, both practically (since it models a buyer and a marketplace as mentioned above) and technically. We first consider monotone DNF formulas, which have long been known to be hard to learn exactly from membership queries alone but easy to learn approximately (in a strong, distribution-free PAC sense) given membership queries and polynomially many random examples of the target formula (Angluin, 1988). We show that monotone DNF formulas are *hard* for preference elicitation, even with demand queries. However, the hardness we show is $2^{\Omega(\sqrt{n})}$-hard rather than the $2^{\Omega(n)}$-hardness one gets for exact learning of monotone DNF. On the other hand, we show $\log(n)$-DNF formulas are *easy* for preference-elicitation, even if the functions have more than polynomially many terms. We also give a number of general statements about when the ability to succeed for one of these goals implies being able to succeed in the others. After comparing elicitation with boolean-valued and real-valued preference functions, we briefly consider the issue of truthfulness in elicitation. We then summarize subsequent related work that has been done since the conference (COLT-03) version of this paper appeared. Finally, we end with two open problems.

1.2 Related Work on Combinatorial Auctions

There is a substantial literature of work on combinatorial auctions. The standard view of these auctions (without the interactive notion of preference elicitation) is that bidders submit bids on bundles in some appropriate bidding language, and then the auctioneer determines who wins what based on these bids. The issue of determining the winners in such auctions, given the bids, is a complex optimization problem that has received considerable

2. In a demand query, the auctioneer proposes a price for each item and then asks the bidder to specify a single subset of the items that is optimal for the bidder given those prices. Demand queries will be discussed further in Section 2.

attention (Rothkopf et al., 1998, Sandholm, 2002a, Fujishima et al., 1999, Nisan, 2000, Andersson et al., 2000, Sandholm et al., 2001). Communication complexity issues have received substantial attention as well. There are $2^n - 1$ bundles, and each agent may need to bid on all of them to fully express its preferences. Appropriate bidding languages (Sandholm, 2002a,b, Fujishima et al., 1999, Nisan, 2000, Hoos and Boutilier, 2001, Sandholm and Suri, 2001) can address the communication overhead in some cases where the bidder's utility function is compressible. However, they still require the agents to completely determine and transmit their valuation functions and as such do not solve all the issues. So in practice, when the number of items for sale is even moderate, the bidders cannot bid on all bundles. Instead, they may bid on bundles which they will not win, and they may fail to bid on bundles they would have won. The former problem leads to wasted effort, and the latter problem leads to reduced economic efficiency of the resulting allocation of items to bidders.

Preference elicitation, the subject of this paper, was recently proposed to address these problems (Conen and Sandholm, 2001), and several papers have studied different types of elicitors (Conen and Sandholm, 2002b,a, Hudson and Sandholm, 2004, Nisan and Segal, 2003, Smith et al., 2002). On the negative side, if valuations are arbitrary monotone functions, then the worst-case communication complexity to find an (even approximately) optimal allocation is exponential in the number of items, no matter what query types are used (Nisan and Segal, 2003). Nonetheless, empirically, preference elicitation can provide a substantial savings compared to explicitly bidding on all $2^n - 1$ bundles (Hudson and Sandholm, 2004).

An issue that arises in the study of auctions is that bidders may lie about their preferences (e.g., bidder f when queried with x may not truthfully report $f(x)$) if they believe it may help in the final allocation. However, *Vickrey-Clarke-Groves (VCG)* schemes (Vickrey, 1961, Clarke, 1971, Groves, 1973) provide a method for charging bidders so that each is motivated to tell the truth about its valuations. Briefly, in this scheme the elicitor first finds the optimal allocation OPT under the assumption that bidders are behaving truthfully. Then, for each bidder i, the elicitor finds the optimal allocation OPT_i without bidder i, again assuming truthful behavior. Finally, bidder i is charged a fee based on the difference between the utility of the other agents in OPT and OPT_i. It seems perhaps circular at first glance, but one can show that in such a scheme, each bidder will in fact be motivated to be truthful throughout the whole process (Conen and Sandholm, 2001). Formally, bidding truthfully is an *ex-post equilibrium*. We discuss these issues in more detail in Section 6, but the conclusion is that if one can elicit the optimal allocation exactly assuming that agents tell the truth, one can determine VCG payments that make truth-telling the best strategy for the bidders. Because of this, for the remainder of the paper (except Section 6) we will assume that the bidders are truthful.

Driven by the same concerns as preference elicitation in combinatorial auctions, there has also been significant recent work on ascending combinatorial auctions (Parkes, 1999b,a, Ausubel and Milgrom, 2002, Wurman and Wellman, 2000, Bikhchandani et al., 2001, Bikhchandani and Ostroy, 2001). These are multistage mechanisms. At each stage the auctioneer announces prices (on items or in some cases on bundles of items), and each bidder states which bundle of items he would prefer (that is, which bundle would maximize his valuation minus the price he would have to pay for the bundle) at those prices. The auctioneer increases the prices between stages, and the auction usually ends when the optimal allocation is found. Ascending auctions can be viewed as a special case of preference elicitation where the queries are demand queries ("If these were the prices, what bundle would you buy from the auction?") and the query policy is constrained to increasing the prices in the queries over time. Recently it was shown that if *per-item* prices suffice to support an optimal allocation (i.e., a *Walrasian equilibrium* exists), then the optimal allocation can be found with a polynomial number of queries, where each query and answer is of polynomial size (Nisan and Segal, 2003).

Recently, some of us (Zinkevich et al., 2003), noticing the connection to query learning, showed how the algorithm of Angluin et al. (1993) for learning read-once formulas over standard boolean gates could be adapted to elicit preferences expressible as read-once-formulas over gates that are especially natural in the context of combinatorial auctions. This work goes on to discuss computational considerations, showing on the negative side that allocation can be NP-hard even for two bidders with read-once-formula preferences, but on the other hand, allocation can be done in polynomial time if one of the two bidders has a linear value function.

2. Notation and Definitions

Because subset notation is most natural from the point of view of preference elicitation, we will use both subset notation and bit-vector notation in this paper. That is, we will think of the instance space X both as elements of $\{0,1\}^n$ and as the power set of some set S of n *items*. We will also interchangeably call a subset of S a "bundle" or an "example". When discussing preference elicitation, we assume there are k bidders with monotone real-valued preference functions over the instance space. The objective of preference elicitation is to determine a k-way partition $(S_1, \ldots, S_k)$ of S to maximize $f_1(S_1) + f_2(S_2) + \ldots + f_k(S_k)$, where $f_1, \ldots, f_k$ are the k real-valued preference functions. Unless otherwise noted, we will assume $k = 2$; Section 1.1 provides some justification for this focus.

Let C be a class of monotone functions. We will be interested in the learnability of various C — that is, in various subsets of the class of monotone functions — in the exact learning, approximate learning, and preference elicitation models given the ability to make various types of queries. While learning algorithms are typically considered efficient if they run in time polynomial in the number of items n and in the length of the representation of the target (and possibly other parameters), we will at times explicitly require run time bounds independent of description length in order to demonstrate a fundamental advantage of preference elicitation for problems involving complex targets. The hardness observations for learning problems when this restriction is in place are therefore not hardness results in the standard learning-theoretic sense.

Query types: A *membership query* or *value query* is a request $x \in \{0,1\}^n$ to an oracle for a target f. The oracle responds with the value $f(x)$ corresponding to x. We can think of these queries as asking the following question of a bidder: "How much are you willing to pay for this bundle of items?"

A *demand query* is a request $w \in (\mathbf{R}^+)^n$ ($\mathbf{R}^+$ here represents non-negative real values) to an oracle for a target f. The oracle responds with an example $x \in \{0,1\}^n$ that maximizes $f(x) - w \cdot x$. We can think of a demand query w as asking the following question of a bidder:

"If you were in a store in which item i had cost w_i, what subset of items would you choose to buy?"

We can illustrate the power of demand queries with the following observation due to Nisan. If one of the bidders has a linear valuation function, and the other is arbitrary, then preference elicitation can be done with $n+1$ queries: n value queries and one demand query. Specifically, we simply ask the linear bidder n value queries to determine his value on each item, and then send the other bidder these values as prices and ask him what he would like to buy. Thus it is interesting that our main lower bounds, for elicitation of monotone DNF formulas, hold for demand queries as well.[3]

Natural function/representation classes: One of the most natural representation classes of monotone functions in machine learning is that of monotone DNF formulas. In combinatorial auctions, the analog of this representation is called the "XOR bidding language" (Sandholm, 2002a).[4] A preference in this representation is a set of bundles (terms) $T = \{T_1, T_2, \ldots, T_m\}$ along with a set of values $v = \{v_1, v_2, \ldots, v_m\}$, one value v_i for each bundle T_i. The value of this preference for all $S' \subseteq S$ is

$$f_{T,v}(S') = \max_{T_i \subseteq S'} v_i.$$

In other words, the value of a set of items S' is the maximum value of any of the "desired bundles" in T that are contained in S'. We will call this the *DNF representation* of preferences, or "DNF preferences" for short. Our hardness results for this class will all go through for the boolean case (all v_i are equal to 1), while two of our positive results will hold for general v_i.

3. DNF Preferences

Angluin (1988) shows that monotone DNF formulas are hard to exactly learn from value (membership) queries alone, but are easy to learn approximately given membership queries plus random examples (in the PAC distribution-free, strong learning model). Angluin's example showing hardness of exact learning can be thought of as follows: imagine the n items are really $n/2$ pairs of shoes. The buyer would be happy with any bundle containing at least one pair of shoes (any such bundle is worth \$1). But then we add one final term to the DNF: a bundle of size $n/2$ containing exactly one shoe from each pair, where for each pair we flip a coin to decide whether to include the left or right shoe. Since the learning algorithm already knows the answer will be positive to any query containing a pair of shoes, the only interesting queries are those that contain no such pair, and therefore it has to match

3. Lahaie and Parkes (2004) recently studied a more powerful notion of demand query in which one can propose an arbitrary polynomial-size function $h(x)$, and receive the x that maximizes $f(x) - h(x)$. With this type of query, one *can* elicit monotone DNF formulas, since one can now directly apply Angluin's algorithm (Angluin, 1988), making a demand query of this kind whenever Angluin's algorithm would make an equivalence query. Lahaie and Parkes go on to explore further the power of this query class.
4. This terminology is to indicate that the bidder wants only one of his listed bundles and will not pay more for a set of items that contains multiple bundles inside it. This usage is very different from the standard definition of XOR as a sum modulo 2. Therefore, to avoid confusion, we will not use the XOR terminology here.

the last term *exactly* to provide any information. Thus even for a randomized algorithm, an expected $2^{n/2-1}$ value queries are needed for exact learning of monotone DNF formulas.

We now consider the two-bidder preference elicitation problem when one or both of the preferences are represented as monotone DNF expressions, beginning with a few simple observations.

Observation 1 *If f is a* known *DNF preference function with m terms, and g is an* arbitrary *unknown monotone preference function, then preference elicitation can be performed using $m+1$ value queries.*

Proof Because g is monotone, the optimal allocation will either be of the form $(T_i, S-T_i)$, for some term T_i in f, or else all of the items in S will go to the bidder with preference g. So, we simply need to query g once for S and once for each set $S-T_i$ and then pick the best of these $m+1$ partitions. ■

Observation 2 *If f and g are boolean DNF preferences each containing at most one term with more than two literals (the hard case in Angluin's construction) then preference elicitation can be performed using poly(n) value queries.*

Proof Since f and g are boolean (they each assign every bundle a value of either 0 or 1), the problem is simply to find terms T_f in f and T_g in g which have no items in common, if such a pair of terms exists. We begin by finding all terms in f of size ≤ 2 by asking n^2 queries. Suppose two of these terms T_1 and T_2 are disjoint. In that case, we query g on $S-T_1$ and $S-T_2$. If one answer is 1 then we are done. If both answers are 0 then this means all of g's terms intersect both T_1 and T_2. In particular, g can have only a constant number of terms, and therefore *exactly* learning g is easy, after which we can then apply Observation 1 (swapping f and g). On the other hand, if f does not have two disjoint terms of size ≤ 2, then the only way f can have more than three such terms is if they all share some common item x_i. It is thus now easy to learn the large term in f: if $f(S-\{x_i\}) = 1$, we can find this term by "walking downward" from the example $S-\{x_i\}$, that is, by iteratively removing as many items as possible from this bundle subject to keeping f's value 1. On the other hand, if $f(S-\{x_i\}) = 0$, meaning that the large term contains x_i as well, we can walk downward from the example in which all the *other* items in the small terms have been removed. Once f has been learned, we can again apply Observation 1. ■

We now show that even though Angluin's specific example is no longer hard in the preference elicitation model, monotone DNF formulas (defining boolean preference functions) remain hard for preference elicitation using value queries, even when the preference functions are quite small. We then extend this result to demand queries as well.

Theorem 1 *Preference elicitation of monotone DNF formulas requires $2^{\Omega(\sqrt{n})}$ value queries. This holds even if each bidder's preference function has only $O(\sqrt{n})$ terms.*

Proof We construct a hard example as follows. There will be $n = m^2$ items, arranged in an m-by-m matrix. Let us label the items x_{ij} for $1 \leq i, j \leq m$. We will call the two preference functions f_R and f_C. Both will be boolean functions. Bidder f_R is happy with any row: that is, $f_R = x_{11}x_{12}\cdots x_{1m} \vee x_{21}x_{22}\cdots x_{2m} \vee \ldots \vee x_{m1}x_{m2}\cdots x_{mm}$. Bidder f_C is happy with any column: that is, $f_C = x_{11}x_{21}\cdots x_{m1} \vee x_{21}x_{22}\cdots x_{m2} \vee \ldots \vee x_{1m}x_{2m}\cdots x_{mm}$. Thus, at this point, it is impossible to make both bidders happy. However, we now add one additional term to each preference function. We flip a coin for each of the n items in S, labeling the item as heads or tails. Let H be the set of all items labeled heads, and T be the set of all items labeled tails. We now add the conjunction of all items in H as one additional term to f_R, and the conjunction of all items in T as one additional term to f_C. Thus now it *is* possible to make both bidders happy, and the optimal allocation will be to give the items in H to the "row bidder" and the items in T to the "column bidder".

We now argue that no query algorithm can find this allocation in less than $\frac{1}{2}2^{\sqrt{n}} - 2$ queries in expectation. Let us enforce that the last two questions of the query protocol are the values of the actual allocation. That is, if the elicitor assigns the items in H to the row agent and T to the column agent, it must ask the row agent the value of H and the column agent the value of T. This constraint only increases the length of the protocol by at most 2 questions.

Let us assume that the elicitor knows in advance the structure of the problem, the row sets and the column sets, and the only information the elicitor does not know are the sets H and T. In this case, we can assume without loss of generality that the elicitor never asks the row bidder about any bundle containing a row (because he already knows the answer will be "yes") and similarly never asks the column bidder about any bundle containing a column.

We now argue as follows. If the elicitor asks a query of the row bidder, the query must be missing at least one item in each row, and if the elicitor ask a query of the column bidder, it must be missing at least one item in each column. However, notice that in the first case, the answer will be positive only if all missing items are in T, and in the second case, the answer will be positive only if all missing items are in H. Therefore, for any given such query, the probability that the answer will be positive taken over the random coin flips is at most $2^{-\sqrt{n}}$. Thus, for any elicitation strategy, the probability the elicitor gets a positive response in the first q queries is at most $q2^{-\sqrt{n}}$ and therefore the expected number of queries is at least $\frac{1}{2}2^{\sqrt{n}}$. ■

We now show that preference elicitation remains hard for DNF preferences even if we allow demand queries.

Theorem 2 *Even if both demand queries and value queries are allowed, preference elicitation of monotone DNF formulas requires* $2^{\Omega(\sqrt{n})}$ *queries. This holds even if each bidder's preference function has only* $O(\sqrt{n})$ *terms.*

Proof We use the same example as in the proof of Theorem 1. As in that proof, we can insist that the last question be a demand query where the agent responds with the set H or T respectively. Let us without loss of generality consider a sequence of demand queries to the "row bidder". What we need to calculate now is the probability, for any given cost vector w, that the set H happens to be the cheapest term in his DNF formula. The intuition is that this is highly unlikely because H is so much larger than the other terms.

Specifically, for a given query cost vector w, let w_i be the total cost of the ith row. Thus, the cheapest row has cost $\min(w_1, \ldots, w_m)$ and the *expected* cost of H is $\frac{1}{2}(w_1 + \ldots + w_m)$. One simple observation that helps in the analysis is that if we define h_i as the cost of the items in H that are in the ith row, then $\Pr(h_i \geq w_i/2) \geq 1/2$. That is because if any particular subset of the ith row has cost less than $w_i/2$, its complement in the ith row must have cost greater than $w_i/2$. Furthermore, these events are independent over the different rows.

So, we can reduce the problem to the following: we have m independent events each of probability at least $1/2$. If at least two of these events occur, the elicitor gets no information (H is not the cheapest bundle because it is not cheaper than the cheapest row). Thus, the probability the elicitor *does* get some information is at most $(m+1)2^{-m}$ and the expected number of queries is at least $\frac{1}{2(m+1)}2^m$. ■

Open Problem 1 *Can preferences expressible as polynomial-size DNF formulas be elicited in $2^{O(\sqrt{n})}$ value queries or demand queries? (This is open even for the boolean preference case.)*

3.1 log(n)-DNF Preferences

In the previous problem, even though there were only $O(\sqrt{n})$ terms in each preference function, the terms themselves were fairly large. What if all of the terms are small, of size no more than $\log n$? Observe that there are $\binom{n}{\log n}$ possible terms of size $\log n$, so some members of this class cannot be represented in $\text{poly}(n)$ bits.

Theorem 3 *If f and g are DNF-preferences where all terms are of $O(\log n)$, then preference elicitation can be performed in a number of value queries polynomial in n.*

Proof We begin by giving a randomized construction and then show a derandomization.

For convenience let us put an empty term T_0 of value 0 into both f and g. With this convention we can assume the optimal allocation satisfies some term $T' \in f$ and some term $T'' \in g$.

We now simply notice that since T' and T'' are both of size $O(\log n)$, a random partition (S', S'') has probability at least $1/\text{poly}(n)$ of having the property that $S' \supseteq T'$ and $S'' \supseteq T''$. So, we simply need to try $\text{poly}(n, \log \frac{1}{\delta})$ random partitions and take the best one, and with probability at least $1 - \delta$ we will have found the optimal allocation.

We can now derandomize this algorithm using the (n, k)-universal sets of Naor and Naor (1990). A set of assignments to n boolean variables is (n, k)-universal if for every subset of k variables, the induced assignments to those variables covers all 2^k possible settings. Naor and Naor (1990) give efficient explicit constructions of such sets using only $2^{O(k)} \log n$ assignments. In our case, we can use the case of $k = O(\log n)$, so the construction is polynomial time and size. Each of these assignments corresponds to a partition of the items, and we simply ask f and g for their valuations on each one and take the best. ■

4. General Relationships

In this section we describe a number of general relationships between query learning and preference elicitation. We will solely concern ourselves here with communication/query complexity issues, and not with the issue of computation time.

To begin with some simple relationships, it is clear that preference elicitation is no harder than exact learning, since one way to perform elicitation is to simply learn each bidder's preferences exactly. On the other hand, the results from the previous section show that this is not true for approximate learning. Occam's razor theorems (Blumer et al., 1987) imply that *any* function can be approximately learned with a number of queries polynomial in the description length of the function (ignoring issues of computation time), and thus Theorems 1 and 2 imply a super-polynomial gap between the number of value queries needed for preference elicitation and approximate learning for the case of monotone DNF formulas. In the other direction, we have seen several examples of cases where preference elicitation is *easy* and yet it is hard to perform exact learning (Theorem 3) or even approximate learning (example in Section 1.1) because the target function cannot be written down, even approximately, in a small number of bits.

These last examples are something of a "cheat" because the function cannot even be described compactly. We now describe a case in which preference elicitation is easy but exact learning is hard, even though the function has a small description. We then show that in certain circumstances, however, the ability to elicit does imply the ability to learn with queries.

4.1 Almost-Threshold Preferences

We now define a class of boolean preference functions that we call *almost-threshold*. This class will be used to show that, even if all of the functions in a class have representations of size polynomial in n, we can still separate exact learning and preference elicitation with respect to value queries. In fact, the gap is super-exponential.

An "almost threshold" preference function is defined by specifying a single set S'. This set in turn defines a preference function that is 1 for any set of size greater than or equal to $|S'|$, except for S' itself, and is 0 otherwise. Formally, for any $S' \neq \emptyset$, define

$$h_{S'}(S'') = \begin{cases} 1 & \text{if } S'' \neq S' \text{ and } |S''| \geq |S'| \\ 0 & \text{otherwise} \end{cases}$$

The class H_{AT} of almost-threshold preference functions is then $H_{AT} = \{h_{S'}\}$.

Observation 3 *It requires at least $\binom{n}{\lceil n/2 \rceil - 1}$ value queries to exactly learn the class H_{AT}.*

Theorem 4 *If $f, g \in H_{AT}$ then the optimal allocation can be elicited in $4 + \log_2 n$ value queries.*

Proof Recall that we use S to represent the set of all items, and assume $|S| > 2$. Also suppose $f = h_{S'}$. The first step is to determine $|S'|$. We can do this in $\log_2 n + 1$ queries using binary search. We next find two sets T, T' of size $|S'|$ such that $f(T) = f(T') = 1$. This can be done by picking three arbitrary sets of size $|S'|$ and querying the first two: at most one of these two sets can have the value 0, and if one of the two sets does have this value then the third set must have the value 1. Our final query is for the value of $g(S-T)$. If this query returns 1, then $T, S-T$ is an optimal allocation (and has value 2). Otherwise, $T', S-T'$, regardless of its value, is an optimal allocation. Thus, we can find the optimal allocation in $4+\log_2 n$ value queries, although we may need one more query if we wish to determine the value of this allocation. ■

4.2 When Easy Elicitation Implies Easy Learning

We now show that in certain circumstances, however, the ability to elicit with a polynomial number of value queries does imply the ability to exactly learn with a similar number of queries. In particular, we will show that if preference elicitation can be performed query-efficiently for preferences drawn from certain boolean classes then a superset query can be efficiently simulated using value queries.

Definition 5 *A* **superset oracle** *for a target f^* in a boolean concept class H takes a function $f \in H$ as input. If f is a superset of f^*, that is, $\{x : f(x) = 1\} \supseteq \{x : f^*(x) = 1\}$, then the query returns "true". Otherwise the query produces a counterexample: an x such that $f(x) = 0$ but $f^*(x) = 1$.*

Recall that Angluin's algorithm (Angluin, 1988) for learning monotone DNF uses value (membership) and equivalence queries, but that the equivalence oracle is always queried with a hypothesis that is a subset of the target function (it is an improper subset, the target itself, on the final query). Therefore, the same algorithm can be used to learn monotone DNF from a value and superset oracle. In fact, it can be seen that any subclass of monotone DNF that is closed under removal of terms can be learned from value and superset queries by the same algorithm.

What makes this interesting is the following relationship between preference elicitation and superset queries. First, for any boolean function f, let us define its "dual"

$$\hat{f}(S') = 1 - f(S - S').$$

Or, in other words, $\hat{f}(x) = \bar{f}(\bar{x})$. Given a boolean hypothesis class H, define $\hat{H} = \{\hat{f} : f \in H\}$. For example, the dual of the class of monotone $\log(n)$-DNF formulas is the class of monotone $\log(n)$-CNF formulas. The set of monotone functions is closed under dual.

Theorem 6 *Let H be a boolean concept class with dual $\hat{H}$. If, given value oracles for any $f \in H$ and $g \in \hat{H}$, the optimal allocation S', S'' can be elicited using M value queries, then a superset query for a target $f^* \in H$ can be simulated using $M+2$ queries to a value oracle for f^*.*

Proof Suppose that one wants to perform a superset query with $g \in H$. First, compute $\hat{g} \in \hat{H}$. Then, perform preference elicitation on $f^*, \hat{g}$. If this procedure returns an allocation satisfying both agents, this means we have an x such that $f^*(x) = 1$ and $\hat{g}(\bar{x}) = 1$. But, $\hat{g}(\bar{x}) = \bar{g}(x)$ so this means that x is a counterexample to the superset query. On the other hand, if the elicitation procedure fails to satisfy both agents then no such x exists, so the superset query can return "true" in this case. ■

Corollary 7 *If H is a subclass of monotone DNF that is closed under removal of terms, and if preference elicitation for $(H, \hat{H})$ can be performed in M queries, then H is exactly learnable from a number of value queries that is polynomial in n, M, and the number of terms in the target monotone DNF.*

5. Boolean-Valued Versus Real-Valued Elicitation

It might seem that eliciting real-valued preference functions would generally be much more difficult than eliciting boolean preferences. In this section, we show that—under certain conditions—the number of value queries necessary for elicitation of real-valued preferences is not that much greater than the number of queries required for eliciting boolean preferences.

First of all, for any real-valued function class H we define a related boolean-valued class as follows. Let $\succ$ be a variable representing either the $>$ or $\geq$ relational operator and let $\mathbf{P}(S)$ represent the power set of S. Given a function $f : \mathbf{P}(S) \to \mathbf{R}^+$ and an $a \in \mathbf{R}$, define $thresh^{\succ}_{f,a} : \mathbf{P}(S) \to \{0, 1\}$ to be a function such that

$$thresh^{\succ}_{f,a}(S') = \begin{cases} 1 & \text{if } f(S') \succ a \\ 0 & \text{otherwise.} \end{cases}$$

A function f is said to project onto a set of boolean-valued functions H' if for all $a \in \mathbf{R}$, $thresh^{\geq}_{f,a}, thresh^{>}_{f,a} \in H'$. A set of functions H is said to project onto a set H' if each $f \in H$ projects onto H'. The **boolean projection** of H is the smallest set of functions that H projects onto.

Now, imagine that f and g, the preferences for two agents, are drawn from H. This problem is closely related to the case where preferences f' and g' are drawn from H', the boolean projection of H. By considering the ranges of the functions in H, it is sometimes clear that the above two problems are equally hard. We begin with the observations that when the functions in H all share a small known domain, it is easy to see that these problems are of similar difficulty. We will show in the first theorem below that if the ranges of the functions in H are small sets, then these two problems are still of similar difficulty. On the other hand, in the second theorem of this section, we will show that there exists a real-valued H with a corresponding boolean projection H' such that exponentially more value queries are required to perform preference elicitation on H than are required to exactly learn H' from value queries.

5.1 Single Small Co-Domain

First of all, suppose that H is such that there exists a small set R_H which is a co-domain for every function $f \in H$. That is, for any $f \in H$, for every $S' \subseteq S$, $f(S') \in R_H$. Observe that if the elicitor knows H, then it knows R_H. In this scenario, it is easy to prove that the elicitation of an optimal allocation for $f, g \in H$ and the elicitation of an optimal allocation for $f', g' \in H'$, where H' is the boolean projection of H, can be performed using similar numbers of value queries.

Before we describe the proof, we define a **threshold query**. Let both $\succ$ and $\succ'$ be variables taking on values in $\{>, \geq\}$. Then a $(a, b, \succ\succ')$ threshold query is defined as follows: Does there exist a set $S' \subseteq S$ such that $f(S') \succ a$ and $g(S - S') \succ' b$? If so, return $(f(S'), g(S - S'), S')$, otherwise return false.

Observation 4 *Suppose class H has boolean projection H' such that the optimal allocation for any $f', g' \in H'$ can be elicited in k value queries. Then a threshold query on any $f, g \in H$ can be performed using at most $k + 2$ value queries.*

Proof Suppose that we are attempting to perform an $(a, b, >\geq)$ threshold query on f and g, and let $f' = thresh^{>}_{f,a}$ and $g' = thresh^{\geq}_{g,b}$. Then the threshold query should return false if and only if no allocation can satisfy both f' and g'. So, we simply perform elicitation on f' and g', using value queries to f and g to simulate value queries to f' and g' respectively, and see if both can be satisfied. If so, we perform two more queries (one each to f and g) to determine the actual value of this allocation. ■

Observation 5 *Suppose class H has boolean projection H', and suppose that R_H is a co-domain for every function $f \in H$. If $|R_H| = m$, and the optimal allocation for two preferences $f', g' \in H'$ can be elicited in k value queries, then the optimal allocation for any two preferences $f, g \in H$ can be elicited in $(k + 2)m^2$ value queries.*

Proof

The algorithm to elicit the optimal allocation for $f, g \in H$ is as follows. For all $(a, b) \in R^2_H$, perform an $(a, b, \geq\geq)$ threshold query, and let X be the set of all non-false responses (a, b, S') returned by these queries. Return the allocation $(S'', S - S'')$ from the triple $(a'', b'', S'') \in X$ that maximizes $a'' + b''$.

By the previous observation, all of these threshold queries can be simulated using at most $(k + 2)m^2$ value queries. To see that this algorithm returns an optimal allocation, let $(S^*, S - S^*)$ be a fixed optimal allocation, and define $f^* = f(S^*)$ and $g^* = g(S - S^*)$. Observe that $(f^*, g^*) \in R^2_H$, so the algorithm will make the threshold query $(f^*, g^*, \geq, \geq)$. Furthermore, this query will return some set S'' such that $f(S'') = f^*$ and $g(S - S'') = g^*$, since such an S'' exists and since by the optimality of S^* any response (a, b, S'') to this query must have $a \leq f^*$ (since $b \geq g^*$) and $b \leq g^*$ (since $a \geq f^*$). Thus X contains at least one optimal allocation. ■

5.2 Many Small Ranges

We again consider a set of real valued preference functions H. However, we will not assume a finite co-domain shared by all of the functions $f \in H$. Instead, we will assume that there exists an m such that for every $f \in H$, the size of the range[5] of f is bounded by m.

5. The size of the range of a function is the number of elements in the range. Technically, for a function $f : X \to Y$, the size of the range is $|\{y \in Y : \exists x \in X \text{ such that } f(x) = y\}|$.

Such problems are not as conducive to the easy analysis of the earlier section. For instance, given a function $f \in H$, we may not be able to discover its entire range without a number of queries exponential in the number of items. However:

Theorem 8 *Given an integer m and a set H such that each function $f \in H$ has a range of size less than m, if H' is the boolean projection of H, and the optimal allocation for $f', g' \in H'$ can be elicited using k value queries, then the optimal allocation for $f, g \in H$ can be elicited using $2 + 4(k+2)m^2$ value queries.*

Proof

Let $(S^*, S - S^*)$ be any fixed optimal allocation, $f^* = f(S^*)$, and $g^* = g(S - S^*)$. We give an algorithm that will iteratively construct subsets of the ranges of f and g such that, when the construction is complete, the final subsets will contain f^* and g^*, respectively. By the analysis of Observation 5, if the algorithm of that observation is run using the cross product of these two subsets in place of R_H^2, the algorithm will still successfully locate an optimal allocation.

The full algorithm is as follows:

1. Initialize $R_f = \{f(\emptyset)\}$ and $R_g = \{g(\emptyset)\}$.
2. For all $(a, b) \in R_f \times R_g$, if one has not already done so, perform three threshold queries $(a, b, >>)$, $(a, b, >\geq)$, and $(a, b, \geq>)$.
3. For every triple (a', b', S') returned in the previous step, add a' to R_f and b' to R_g.
4. If R_f and R_g increased in size on the previous step, return to step 2.
5. If R_f and R_g did not increase in size, run the algorithm of Observation 5 using $R_f \times R_g$ in place of R_H^2.

Let the final values of R_f and R_g be denoted by R_f^* and R_g^*, respectively. Observe that R_f^* is a subset of the range of f, because only values of f are inserted into it. Similarly, R_g^* is a subset of the range of g. Thus $|R_f^* \times R_g^*| \leq m^2$. Observe that 2 value queries are made in step 1, $3|R_f^* \times R_g^*|$ threshold queries are made in step 2 (which, by Observation 4, can be simulated by at most $3(k+2)m^2$ value queries), and, by Observation 5 and the earlier analysis, no more than $(k+2)m^2$ value queries are made in step 5. Thus, no more than $2 + 4(k+2)m^2$ value queries are made.

Now, consider the sets $R_f \subseteq R_f^*$ and $R_g \subseteq R_g^*$ just before the ith execution of step 2 of the algorithm. We will show below that if at least one of f^* and g^* is not contained in R_f and R_g, respectively, then at least one of these sets will increase in size when this step is executed. Thus, the algorithm will continue iterating this step until $f^* \in R_f$ and $g^* \in R_g$.

First, assume that both $f^* \notin R_f$ and $g^* \notin R_g$, and let $f^<$ $(g^<)$ be the largest value in R_f (R_g) that is less than f^* (g^*). Note that the value $f^<$ exists[6] because $f(\emptyset) \in R_f$ and for all S', $f(\emptyset) \leq f(S')$ by the monotonicity of preference functions. Similarly, $g^<$ exists. Therefore, at some iteration of the algorithm the threshold query $(f^<, g^<, >>)$ will be made, and when it is made it will not return "false" because (f^*, g^*, S^*) is a valid response.

6. If $f(\emptyset) = f^*$, then $f^* \in R_f$ at all stages.

Furthermore, any response (a, b, S') must either have $a \leq f^*$ or $b \leq g^*$, since $a+b \leq f^*+g^*$. But, by the definition of $f^<$ and $g^<$ as well as of the threshold query, this means that at least one of a or b is a value that is not contained in R_f or R_g. But both a and b will be contained in their respective R^* sets. Therefore, at least one of the conditions $R_f \neq R_f^*$ and $R_g \neq R_g^*$ holds, so at least one of the sets R_f and R_g must grow during execution i of step 2.

Next, consider the case when $f^* \in R_f$ and $g^* \notin R_g$ (the remaining case $f^* \notin R_f$ and $g^* \in R_g$ is symmetric). Define $g^<$ as above and consider the threshold query $(f^*, g^<, \geq >)$. Reasoning as above shows that this query will produce a response (a, b, S') such that $b \notin R_g$. Thus, in all cases when at least one of f^* or g^* is not in its respective set, one of the sets grows at step 2.

■

5.3 When Real Values Make a Problem More Difficult

Theorem 9 *There exists a class of real-valued functions that requires $2^n - 1$ value queries to elicit while its boolean projection requires at most $n + 1$ value queries to exactly learn.*

Proof

Imagine that the items are "more or less" unrelated. In particular, each item has a basic value in $\{1, 2, 4, 8, \ldots, 2^{n-1}\}$. For all $a \in S$, define $V(a)$ to be the basic value of a, and assume that this mapping is known. For all $S' \subseteq S$, define $V(S') = \sum_{a \in S'} V(a)$. Thus, the basic value of any set is the sum of the basic values in that set. Observe that for any $S' \subseteq S$, $V(S') + V(S - S') = 2^n - 1$.

Now, each agent has a special set that they value slightly more than other agents do. Thus, for all $S', S'' \subseteq S$, define $f_{S'}(S'') = V(S'')$ if $S' \neq S''$, and $f_{S'}(S') = V(S') + \frac{1}{2}$. The class of preferences of interest is therefore $\{f_{S'}\}_{S' \subseteq S}$.

In order to determine the optimal allocation when preferences are drawn from this class, the elicitor must find the special set for one agent. And in the worst case it requires $2^n - 1$ value queries in order to find a special set, since the only information obtained from a value query on a non-special set is that it is not special. Therefore, $2^n - 1$ value queries are required to elicit this preference class.

Now, consider the boolean projection of this class. For all $a \in R$, for all $S' \subseteq S$, define $g_a(S')$ to be true if and only if $V(S') \geq a$. Then the projection can be represented as $\{g_a\}_{a \in \{0,\ldots,2^n\}}$. Now, using value queries, we can perform a binary search for the value of a defining a target member of this class. Thus, we can exactly learn a target g in the boolean projection with at most $n + 1$ value queries. ■

One point to observe is that it is easy to approximate any function with an exponential number of values with a function with a polynomial number of values. However, one must be careful when one computes an allocation that is only approximately optimal, because the traditional techniques to motivate the agents to answer truthfully (which we describe in the next section) will no longer work.

6. Truthfulness and VCG

In combinatorial auctions and mechanism design, one key issue that arises is that bidders have their own interests: they each want to receive as valuable a bundle as possible, and therefore may lie in their responses if they perceive it to be to their advantage. For example, if the auctioneer is not going to actually charge the bidders anything for the bundles they get, then bidders have an incentive to report overly high valuations, in order to make the auctioneer think that social welfare will be improved by giving more to them. On the other hand, if the bidders are charged exactly the valuations that they report, then they have an incentive to underbid, in the hope of making a profit (paying less for a bundle than it is actually worth to them).[7] In preference elicitation, the issue of motivating the bidders to answer queries truthfully is exacerbated by the fact that the elicitor's queries leak information to the bidder about the answers that other bidders have given.

Recently, a methodology was proposed by which elicitors can be made incentive compatible in the sense that every bidder answering the queries truthfully is an *ex post equilibrium* (Conen and Sandholm, 2001).[8] This is accomplished by organizing the mechanism so that if all the bidders answer truthfully, the final allocation and payments follow the *Vickrey-Clarke-Groves scheme (VCG)* (Vickrey, 1961, Clarke, 1971, Groves, 1973). In this scheme (the Clarke version), the amount bidder i has to pay is the sum of others' revealed valuations for the bundles they get had bidder i not participated, minus the sum of others' revealed valuations for the bundles they get in the actual optimal allocation. The elicitor can determine these payments by asking enough queries to be able to determine the welfare maximizing allocation overall, and *by asking extra queries to determine the welfare maximizing allocation for the auctions where each agent is ignored in turn.* The essence of the argument is that the auction in which agent i is removed serves only to determine agent i's payment, and therefore in this auction there is no motivation for any of the participating agents to lie. This then means that the payments given to the bidders can be assumed to be the correct VCG payments, which then implies by standard VCG arguments that the optimal strategy for the bidders in the first auction is to tell the truth as well. Conceptually, one could think of $k+1$ "elicitors", each working to solve one of these problems. Once all of the "elicitors" have found their welfare maximizing allocations respectively, the process can terminate. Note that the extra overhead of motivating the bidders to bid truthfully is just

7. Throughout this paper we have let each bidder i have some valuation function f_i from bundles of items to reals. This notation implicitly makes the following common economic assumptions: (1) *private values*: bidder i knows f_i (in other words, the *function* f_i does not depend on the other bidders in any way); and (2) *no externalities*: bidder i does not care who gets the items that i does not get. For the truthfulness discussion we additionally make the common economic assumption (3) *quasilinear preferences*: the utility that bidder i tries to maximize is $u_i(S_i, p_i) = f_i(S_i) - p_i$, where S_i is the set of items that i gets and p_i is the total price that i has to pay.
8. This means that bidding truthfully is each bidder's best strategy (for any prior probability distribution that he may hold about the other bidders) given that the other bidders bid truthfully. In other words, truthful bidding strategies form a Nash equilibrium even in hindsight. This does not mean that bidding truthfully is a dominant strategy: if others bid insincerely, one may also do better by bidding insincerely. For example, in a 2-bidder setting, if bidder 1's strategy involves dropping out (bidding zero from then on) whenever it receives a particular query stream, then it can be bidder 2's best strategy to answer queries in a way that causes the elicitor to submit that query stream to bidder 1. In summary, implementation in *ex post* equilibrium is stronger than implementation in Nash equilibrium, but weaker than implementation in dominant strategies.

solving k additional elicitation problems beyond the original elicitation problem. Therefore, if elicitation can be done in a polynomial number of queries, then so can elicitation that motivates the bidders to answer the queries truthfully.

7. Subsequent Work

Since the conference (COLT-03) version of this paper appeared, a significant amount of closely related work has been done. In this section we summarize that work.

One of the oldest techniques for preference elicitation is an ascending auction. An ascending auction can be considered to be a sequence of increasing demand queries, where if one asks a query w' after a query w, then it must be the case that for all i, $w'_i \geq w_i$. In the conference (COLT-03) version of this paper we presented the following problem as an interesting open question:

Open Problem 2 *Does there exist a preference elicitation problem that is hard (or impossible) to elicit using an ascending auction but easy to elicit using demand queries?*

Since then, this question has been answered, and the answer is affirmative. Nisan (2003) presents a 2-item auction where no ascending item-price auction can determine the optimal allocation (using *any* number of queries), but the optimal allocation can easily be determined using (nonascending) item-price demand queries.

On the other hand, if *bundle-price* demand queries are allowed — i.e., prices are not assigned to items only, but potentially also to bundles — then ascending auctions exist that always determine the optimal allocation (using potentially an exponential number of queries), at least if each bidder is assumed to act truthfully (Parkes and Ungar, 2002, Ausubel and Milgrom, 2002). As pointed out by Nisan (2003), it remains an open question whether there exists an ascending bundle-price auction that always determines the optimal allocation if the auction is restricted to being *anonymous*, that is, at any time, the price of a bundle is the same for each agent. Bundle-price demand queries are quite powerful: as mentioned in footnote 3, one can use them to efficiently elicit monotone DNF formulas, and this fact as well as other results on these queries are given by Lahaie and Parkes (2004).

As to preference elicitation using value queries only, new valuation classes learnable in a polynomial number of queries have been introduced in Conitzer et al. (2003) and Santi et al. (2004). These include valuations where items have at most k-wise dependencies, and certain other valuations. Furthermore, if two classes of valuations are each learnable in a polynomial number of queries, then so is their union—even though the elicitor does not know in advance in which of the two classes (or both) the bidder's valuation belongs. Santi et al. (2004) also present severely restricted valuation classes where learning nevertheless requires an exponential number of value queries. First steps toward a characterization of polynomial learnability of valuation functions are also given.

8. Conclusions and Open Problems

In machine learning, one's objective is nearly always to learn or approximately learn some target function. In this paper, we relate this to the notion of preference elicitation, in which the goal instead is to find the optimal partitioning of some set of items among the bidders.

In the case of two bidders, preference elicitation can be thought of as a learning problem with *two* target functions f and g, where the goal is rather than necessarily learning f and g to instead find the example x that maximizes $f(x) + g(\bar{x})$.

We now describe several open problems left by this work. We begin with a problem stated in Section 3.

Open Problem 2 *Can preferences expressible as polynomial-size DNF formulas be elicited in $2^{O(\sqrt{n})}$ value queries or demand queries?*

A somewhat fuzzier question related to our results on $\log(n)$-DNF is the following. Our algorithm in this case was non-adaptive: the questions asked did not depend on answers to previous questions. It seems natural that for some classes adaptivity should help. In fact, it is not hard to generate artificial examples in which this is the case. However, we know of no natural example having this property.

Open Problem 3 *Are there natural classes of functions for which exact learning is information-theoretically hard, preference elicitation via a non-adaptive algorithm is hard (i.e., an algorithm in which the questions can all be determined in advance) but elicitation by an adaptive algorithm is easy.*

Acknowledgments

This material is based upon work supported under NSF grants CCR-0105488, ITR CCR-0122581, CCR-0209064, ITR IIS-0081246, and ITR IIS-0121678. Any opinion, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation. Tuomas Sandholm is also supported by a Sloan Fellowship. We would like to thank the referees for their helpful comments.

References

Arne Andersson, Mattias Tenhunen, and Fredrik Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, pages 39–46, Boston, MA, 2000.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning read-once formulas with queries. In *Journal of the ACM*, volume 40, pages 185–210, 1993.

Lawrence M. Ausubel and Paul Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1, 2002. No. 1, Article 1.

Sushil Bikhchandani, Sven de Vries, James Schummer, and Rakesh V. Vohra. Linear programming and Vickrey auctions, 2001. Draft.

Sushil Bikhchandani and Joseph M. Ostroy. The package assignment model. UCLA Working Paper Series, mimeo, 2001.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, April 1987.

Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 256–259, Tampa, FL, October 2001. A more detailed description of the algorithmic aspects appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.

Wolfram Conen and Tuomas Sandholm. Differential-revelation VCG mechanisms for combinatorial auctions. In *AAMAS-02 workshop on Agent-Mediated Electronic Commerce (AMEC)*, Bologna, Italy, 2002a. Springer Lecture Notes in Computer Science LNCS 2531.

Wolfram Conen and Tuomas Sandholm. Partial-revelation VCG mechanism for combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 367–372, Edmonton, Canada, 2002b.

Vincent Conitzer, Tuomas Sandholm, and Paolo Santi. Combinatorial auctions with k-wise dependent valuations, October 2003. Draft.

Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

Holger Hoos and Craig Boutilier. Bidding languages for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217, Seattle, WA, 2001.

Benoit Hudson and Tuomas Sandholm. Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *International Conference on Autonomous Agents and Multi-Agent Systems*, New York, NY, USA, 2004.

Sebastién Lahaie and David Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, New York, NY, 2004.

Daniel Lehmann, Lidian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proc. 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, Baltimore, 1990.

Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 1–12, Minneapolis, MN, 2000.

Noam Nisan. The power and limitations of item price combinatorial auctions, 2003. Slides from the FCC Combinatorial Bidding Conference, Queenstown, MD, Nov. 21–23.

Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 242–252, Minneapolis, MN, 2000.

Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting Lindahl prices, 2003. Working Paper (version: March 2003).

David C. Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 148–157, Denver, CO, November 1999a.

David C. Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999b.

David C. Parkes and Lyle Ungar. An ascending-price generalized Vickrey auction, 2002. Draft, Jun.

Michael H. Rothkopf, Aleksandar Pekeč, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002a.

Tuomas Sandholm. eMediator: A next generation electronic commerce server. *Computational Intelligence*, 18(4):656–676, 2002b.

Tuomas Sandholm and Subhash Suri. Side constraints and non-price attributes in markets. In *IJCAI-2001 Workshop on Distributed Constraint Reasoning*, pages 55–61, Seattle, WA, 2001.

Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1102–1108, Seattle, WA, 2001.

Paolo Santi, Vincent Conitzer, and Tuomas Sandholm. Towards a characterization of polynomial preference elicitation with value queries in combinatorial auctions. In *Conference on Learning Theory (COLT)*, Banff, Alberta, Canada, 2004.

Trey Smith, Tuomas Sandholm, and Reid Simmons. Constructing and clearing combinatorial exchanges using preference elicitation. In *AAAI-02 workshop on Preferences in AI and CP: Symbolic Approaches*, pages 87–93, 2002.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

Peter R. Wurman and Michael P. Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 21–29, Minneapolis, MN, October 2000.

Martin Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 176–185, San Diego, CA, 2003.