Towards a Characterization of Polynomial Preference Elicitation with Value Queries in Combinatorial Auctions[⋆] (Extended Abstract)

Paolo Santi[1][⋆⋆], Vincent Conitzer[2], and Tuomas Sandholm[2]

[1] Istituto di Informatica e Telematica, Pisa, 56124, Italy,
Email: `paolo.santi@iit.cnr.it`
[2] Dept. of Computer Science, Carnegie Mellon University
5000 Forbes Avenue Pittsburgh, PA 15213.
Emails: {`conitzer, sandholm`}`@cs.cmu.edu`

Abstract. Communication complexity has recently been recognized as a major obstacle in the implementation of combinatorial auctions. In this paper, we consider a setting in which the auctioneer (elicitor), instead of passively waiting for the bids presented by the bidders, elicits the bidders' preferences (or valuations) by asking value queries. It is known that in the more general case (no restrictions on the bidders' preferences) this approach requires the exchange of an exponential amount of information. However, in practical economic scenarios we might expect that bidders' valuations are somewhat structured. In this paper, we consider several such scenarios, and we show that polynomial elicitation in these cases is often sufficient. We also prove that the family of "easy to elicit" classes of valuations is closed under union. This suggests that efficient preference elicitation is possible in a scenario in which the elicitor, contrary to what it is commonly assumed in the literature on preference elicitation, does not exactly know the class to which the function to elicit belongs. Finally, we discuss what renders a certain class of valuations "easy to elicit with value queries".

1 Introduction

Combinatorial auctions (CAs) have recently emerged as a possible mechanism to improve economic efficiency when many items are on sale. In a CA, bidders can present bids on bundle of items, and thus may easily express complementarities (i.e., the bidder values two items together more than the sum of the valuations of the single items), and substitutabilities (i.e., the two items together are worth less than the sum of the valuations of the single items) between the objects

⋆ This work is supported in part by NSF under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

⋆⋆ This work was done when the author was visiting the Dept. of Computer Science, Carnegie Mellon University.

on sale[3]. CAs can be applied, for instance, to sell spectrum licenses, pollution permits, land lots, and so on [9].

The implementation of CAs poses several challenges, including computing the optimal allocation of the items (also known as the winner determination problem), and efficiently communicating bidders' preferences to the auctioneer.

Historically, the first problem that has been addressed in the literature is winner determination. In [16], it is shown that solving the winner determination problem is NP-hard; even worse, finding a $n^{1/2-\epsilon}$-approximation (here, n is the number of bidders) to the optimal solution is NP-hard [18]. Despite these impossibility results, recent research has shown that in many scenarios the average-case performance of both exact and approximate winner determination algorithms is very good [4, 13, 17, 18, 22]. This is mainly due to the fact that, in practice, bidders' preferences (and, thus, bids) are somewhat structured, where the bid structure is usually induced by the economic scenario considered.

The communication complexity of CAs has been addressed only more recently. In particular, *preference elicitation*, where the auctioneer is enhanced by elicitor software that incrementally elicits the bidders' preferences using queries, has recently been proposed to reduce the communication burden. Elicitation algorithms based on different type of queries (e.g., rank, order, or value queries) have been proposed [6, 7, 12]. Unfortunately, a recent result by Nisan and Segal [15] shows that elicitation algorithms in the worst case have no hope of considerably reducing the communication complexity, because computing the optimal allocation requires the exchange of an exponential amount of information between the elicitor and the bidders. Indeed, the authors prove an even stronger negative result: obtaining a better approximation of the optimal allocation than that generated by auctioning off all objects as a bundle requires the exchange of an exponential amount of information. Thus, the communication burden produced by *any* combinatorial auction design that aims at producing a non-trivial approximation of the optimal allocation is overwhelming, unless the bidders' valuation functions display some structure. This is a far worse scenario than that occurring in single item auctions, where a good approximation to the optimal solution can be found by exchanging a very limited amount of information [3].

For this reason, elicitation in restricted classes of valuation functions has been studied [2, 8, 15, 21]. The goal is to identify classes of valuation functions that are general (in the sense that they allow to express super-, or sub-additivity, or both, between items) and can be elicited in polynomial time.

Preference elicitation in CAs has recently attracted significant interest from machine learning theorists in general [6, 21], and at COLT in particular [2].

1.1 Full elicitation with value queries

In this paper, we consider a setting in which the elicitor's goal is *full* elicitation, i.e., learning the entire valuation function of all the bidders. This definition should be contrasted with the other definition of preference elicitation, in which

[3] In this paper, we will use also the terms super- and sub-additivity to refer complementarities and substitutabilities, respectively.

the elicitor's goal is to elicit enough information from the bidders so that the optimal allocation can be computed. In this paper, we call this type of elicitation *partial* elicitation. Note that, contrary to the case of partial elicitation, in full elicitation we can restrict attention to learning the valuation of a single bidder.

One motivation for studying full elicitation is that, once the full valuation functions of all the bidders are known to the auctioneer, the VCG payments [5, 11, 20] can be computed without further message exchange. Since VCG payments prevent strategic bidding behavior [14], the communication complexity of full preference elicitation is an upper bound to the communication complexity of truthful mechanisms for combinatorial auctions.

In this paper, we focus our attention on a restricted case of full preference elicitation, in which the elicitor can ask only *value queries* (what is the value of a particular bundle?) to the bidders. Our interest in value queries is due to the fact that, from the bidders' point of view, these queries are very intuitive and easy to understand. Furthermore, value queries are in general easier to answer than, for instance, demand (given certain prices for the items, which would be your preferred bundle?) or rank (which is your most valuable bundle?) queries.

Full preference elicitation with value queries has been investigated in a few recent papers. In [21], Zinkevich et al. introduce two classes of valuation functions (read-once formulas and ToolboxDNF formulas) that can be elicited with a polynomial number of value queries. Read-once formulas can express both sub- and super-additivity between objects, while ToolboxDNF formulas can only express super-additive valuations. In [8], we have introduced another class of "easy to elicit with value queries" functions, namely k-wise dependent valuations. Functions in this class can display both sub- and super-additivity, and in general are not monotone[4] (i.e., they can express costly disposal).

1.2 Our contribution

The contributions of this paper can be summarized as follows:

• We introduce the *hypercube representation* of a valuation function, which makes the contribution of every sub-bundle to the valuation of a certain bundle S explicit. This representation is a very powerful tool in the analysis of structural properties of valuations.

• We study several classes of "easy to elicit with value queries" valuations. Besides considering the classes already introduced in the literature, we introduce several new classes of polynomially elicitable valuations.

• We show that the family of "easy to elicit" classes of valuations is closed under union. More formally, we prove that, if $\mathbf{C_1}$ and $\mathbf{C_2}$ are classes of valuations elicitable asking at most $p_1(m)$ and $p_2(m)$ queries, respectively, then any function in $\mathbf{C_1} \bigcup \mathbf{C_2}$ is elicitable asking at most $p_1(m) + p_2(m) + 1$ queries. Furthermore, we prove that this bound cannot be improved.

[4] A valuation function f is *monotone* if $f(S) \geq f(S')$, for any $S' \subseteq S$. This property is also know as *free disposal*, meaning that bidders that receive extra items incur no cost for disposing them.

• The algorithm used to elicit valuations in $\mathbf{C_1} \bigcup \mathbf{C_2}$ might have super-polynomial running time (but asks only polynomially many queries). The question of whether a general polynomial *time* elicitation algorithm exists remains open. However, we present a polynomial time elicitation algorithm which, given any valuation function f in $\mathbf{RO_{+M}} \bigcup \mathbf{Tool_{-t}} \bigcup \mathbf{Tool_t} \bigcup \mathbf{G_2} \bigcup \mathbf{INT}$ (see Section 3 for the definition of the various classes of valuations), learns f correctly. This is an improvement over existing results, in which the elicitor is assumed to know exactly the class to which the valuation function belongs.

• In the last part of the paper, we discuss what renders a certain class of valuations "easy to elicit" with value queries. We introduce the concept of *strongly non-inferable set* of a class of valuations, and we prove that if this set has super-polynomial size then efficient elicitation is not possible. On the other hand, even classes of valuations with empty strongly non-inferable set can be hard to elicit. Furthermore, we introduce the concept of non-deterministic poly-query elicitation, and we prove that a class of valuations is non-deterministically poly-query elicitable if and only if its *teaching dimension* is polynomial.

Overall, our results seem to indicate that, despite the impossibility result of [15], efficient and truthful CA mechanisms are a realistic goal in many economic scenarios. In such scenarios, elicitation can be done using only a simple and very intuitive kind of query, i.e. value query.

2 Preliminaries

Let I denotes the set of items on sale (also called the *grand bundle*), with $|I| = m$. A *valuation function* on I (*valuation* for short) is a function $f : 2^I \mapsto \mathbb{R}^+$ that assigns to any bundle $S \subseteq I$ its valuation. A valuation is *linear*, denoted f_l, if $f_l(S) = \sum_{a \in S} f(a)$. To make the notation less cumbersome, we will use $a, b, \ldots$ to denote singletons, $ab, bc, \ldots$ to denote two-item bundles, and so on.

Given any bundle S, $q(S)$ denotes the value query correspondent to S. In this paper, value queries are the only type of queries the elicitor can ask the bidder in order to learn her preferences. Unless otherwise stated, in the following by "query" we mean "value query".

Definition 1 (PQE) *A class of valuations* $\mathbf{C}$ *is said to be poly-query (fully) elicitable if there exists an elicitation algorithm which, given as input a description of* $\mathbf{C}$*, and by asking value queries only, learns any valuation* $f \in \mathbf{C}$ *asking at most* $p(m)$ *queries, for some polynomial* $p(m)$*. PQE is the set of all classes* $\mathbf{C}$ *that are poly-query elicitable.*

The definition above is concerned only with the number of queries asked (communication complexity). Below, we define a stronger notion of efficiency, accounting for the computational complexity of the elicitation algorithm.

Definition 2 (PTE) *A class of valuations* $\mathbf{C}$ *is said to be poly-time (fully) elicitable if there exists an elicitation algorithm which, given as input a description of* $\mathbf{C}$*, and by asking value queries only, learns any valuation* $f \in \mathbf{C}$ *in polynomial time. PTE is the set of all classes* $\mathbf{C}$ *that are poly-time elicitable.*

It is clear that poly-time elicitability implies poly-query elicitability.

Throughout this paper, we will make extensive use of the following representation of valuation functions. We build the undirected graph H_I introducing a node for any subset of I (including the empty set), and an edge between any two nodes S_1, S_2 such that $S_1 \subset S_2$ and $|S_1| = |S_2| + 1$ (or vice versa). It is immediate that H_I, which represents the lattice of the inclusion relationship between subsets of I, is a binary hypercube of dimension m. Nodes in H_I can be partitioned into levels according to the cardinality of the corresponding subset: level 0 contains the empty set, level 1 the m singletons, level 2 the $\frac{m(m-1)}{2}$ subsets of two items, and so on.

The valuation function f can be represented using H_I by assigning a weight to each node of H_I as follows. We assign weight 0 to the empty set[5], and weight $f(a)$ to any singleton a. Let us now consider a node at level 2, say node ab[6]. The weight of the node is $f(ab) - (f(a) + f(b))$. At the general step i, we assign to node S_1, with $|S_1| = i$, the weight $f(S_1) - \sum_{S \subset S_1} w(S)$, where $w(S)$ denotes the weight of the node corresponding to subset S. We call this representation of f the *hypercube representation* of f, denoted $H_I(f)$.

The hypercube representation of a valuation function makes it explicit the fact that, under the common assumption of no externalities[7], the bidder's valuation of a bundle S depends only on the valuation of all the singletons $a \in S$, and on the relationships between all possible sub-bundles included in S. In general, an arbitrary sub-bundle of S may show positive or negative interactions between the components, or may show no influence on the valuation of S. In the hypercube representation, the contribution of any such sub-bundle on the valuation of S is isolated, and associated as a weight to the corresponding node in H_I.

Given the hypercube representation $H_I(f)$ of f, the valuation of any bundle S can be obtained by summing up the weights of all the nodes S' in $H_I(f)$ such that $S' \subseteq S$. These are the only weights contained in the sub-hypercube of $H_I(f)$ "rooted" at S.

Proposition 1 *Any valuation function f admits a hypercube representation, and this representation is unique.*

Proof. For the proof of this proposition, as well as of all for the proofs of the other theorems presented in this work, see the full version of the paper [19].

Given Proposition 1, the problem of learning f can be equivalently restated as the problem of learning all the weights in $H_I(f)$. In this paper, we will often state the elicitation problem in terms of learning the weights in $H_I(f)$, rather than the value of bundles.

[5] That is, we assume that the valuation function is normalized.

[6] Slightly abusing the notation, we denote with ab both the bundle composed by the two items a and b, and the corresponding node in H_I.

[7] With no externalities, we mean here that the bidder's valuation depends only on the set of items S that she wins, and not on the identity of the bidders who get the items not in S.

Since the number of nodes in H_I is exponential in m, the hypercube representation of f is not compact, and cannot be used directly to elicit f. However, this representation is a powerful tool in the analysis of structural properties of valuation functions.

3 Classes of valuations in PTE

In this section, we consider several classes of valuation functions that can be elicited in polynomial time using value queries.

3.1 Read-once formulas

The class of valuation functions that can be expressed as read-once formulas, which we denote **RO**, has been introduced in [21]. A read-once formula is a function that can be represented as a "reverse" tree, where the root is the output, the leaves are the inputs (corresponding to items), and internal nodes are gates. The leaf nodes are labeled with a real-valued multiplier. The gates can be of the following type: SUM, MAX_c, and ATLEAST_c. The SUM operator simply sums the values of its inputs; the MAX_c operator returns the sum of the c highest inputs; the ATLEAST_c operator returns the sum of its inputs if at least c of them are non-zero, otherwise returns 0. In [21], it is proved that read-once formulas are in PTE.

In general, valuation functions in **RO** can express both complementarities (through the ATLEAST_c operator) and substitutabilities (through the MAX_c operator) between items. If we restrict our attention to the class of read-once formulas that can use only SUM and MAX operators (here, MAX is a shortcut for MAX_1), then only sub-additive valuations can be expressed. This restricted class of read-once formulas is denoted $\mathbf{RO}_{+\mathbf{M}}$ in the following.

3.2 k-wise dependent valuations

The class of k-wise dependent valuations, which we denote $\mathbf{G_k}$, has been defined and analyzed in [8]. k-wise dependent valuations are defined as follows:

Definition 3 *A valuation function f is k-wise dependent if the only mutual interactions between items are on sets of cardinality at most k, for some constant $k > 0$. In other words, the $\mathbf{G_k}$ class corresponds to all valuation functions f such that the weights associated to nodes at level i in $H_I(f)$ are zero whenever $i > k$.*

Note that functions in $\mathbf{G_k}$ might display both sub and super-additivity between items. Furthermore, contrary to most of the classes of valuation functions described so far, k-wise dependent valuations might display costly disposal.

In [8], it is shown that valuations in $\mathbf{G_k}$ can be elicited in polynomial time asking $O(m^k)$ value queries.

3.3 The $\mathbf{Tool_t}$ class

The class of ToolboxDNF formulas, which we denote $\mathbf{Tool_t}$, has been introduced in [21], and is defined as follows:

Definition 4 *A function f is in $\mathbf{Tool_t}$, where t is polynomial in m, if it can be represented by a polynomial p composed of t monomials (minterms), where each monomial is positive.*

For instance, polynomial $p = 3a + 4ab + 2bc + cd$ corresponds to the function which gives value 3 to item a, 0 to item b, value 9 to the bundle abc, and so on. Note if $f \in \mathbf{Tool_t}$, the only non-zero weights in $H_I(f)$ are those associated to the minterms of f.

ToolboxDNF valuations can express only substitutability-free valuations[8], and can be elicited in polynomial time asking $O(mt)$ value queries [21].

3.4 The $\mathbf{Tool_{-t}}$ class

This class of valuation functions is a variation of the ToolboxDNF class introduced in [21]. The class is defined as follows.

Definition 5 $\mathbf{Tool_{-t}}$ *is the class of all the valuation functions f such that exactly t of the weights in $H_I(f)$ are non-zero, where t is polynomial in m. Of these weights, only those associated to singletons can be positive. The bundles associated to non-zero weights in $H_I(f)$ are called the minterms of f.*

In other words, the $\mathbf{Tool_{-t}}$ class corresponds to all valuation functions that can be expressed using a polynomial p with t monomials (minterms), where the only monomials with positive sign are composed by one single literal. For instance, function f defined by $p = 10a + 15b + 3c - 2ab - 3bc$ gives value 10 to item a, value 23 to the bundle ab, and so on.

Theorem 1 *If $f \in \mathbf{Tool_{-t}}$, where t is polynomial in m, then it can be elicited in polynomial time by asking $O(mt)$ queries.*

3.5 Interval valuation functions

The class of interval valuations is inspired by the notion of interval bids [16, 17], which have important economic applications. The class is defined as follows. The items on sale are ordered according to a linear order, and they can display super-additive valuations when bundled together only when the bundle corresponds to an interval in this order. We call this class of sustitutability-free valuations INTERVAL, and we denote the set of all valuations in this class as **INT**.

An example of valuation in **INT** is the following: there are three items on sale, a, b and c, and the linear order is $a < b < c$. We have $f(a) = 10$, $f(b) = 5$,

[8] A valuation function f is substitutability-free if and only if, for any $S_1, S_2 \subseteq I$, we have $f(S_1) + f(S_2) \leq f(S_1 \bigcup S_2)$.

$f(c) = 3$, $f(ab) = 17$, $f(bc) = 10$, $f(ac) = f(a) + f(c) = 13$ (because bundle ac is not an interval in the linear order), and $f(abc) = 21$.

The **INT** class displays several similarities with the $\mathbf{Tool_t}$ class: there are a number of basic bundles (minterms) with non-zero value, and the value of a set of items depends on the value of the bundles that the bidder can form with them. However, the two classes turn out to be not comparable with respect to inclusion, i.e. there exist valuation functions f, f' such that $f \in \mathbf{Tool_t} - \mathbf{INT}$ and $f' \in \mathbf{INT} - \mathbf{Tool_t}$. For instance, the valuation function corresponding to the polynomial $p = a+b+c+ab+bc+ac$ is in $\mathbf{Tool_t} - \mathbf{INT}$, since objects can be bundled "cyclically". On the other hand, the valuation function f of the example above cannot be expressed using a ToolboxDNF function. In fact, the value of the bundles a, b, c, ab, bc and ac gives the polynomial $p' = 10a + 5b + 3c + 2ab + 2bc$. In order to get the value 21 for the bundle abc, which clearly include all the sub-bundles in p', we must add the term abc in p' with *negative* weight -1. Since only positive terms are allowed in $\mathbf{Tool_t}$, it follows that $f \in \mathbf{INT} - \mathbf{Tool_t}$.

What about preference elicitation with value queries in case $f \in \mathbf{INT}$? It turns out that the efficiency of elicitation depends on what the elicitor knows about the linear ordering of the objects. We distinguish three scenarios:

a) the elicitor knows the linear ordering of the items;

b) the elicitor does not know the linear ordering of the items, but the valuation function f to be elicited is such that $f(ab) > f(a) + f(b)$ if and only if a and b are immediate neighbors in the ordering.

c) the elicitor does not know the linear ordering of the items, and the valuation function to be elicited is such that $f(ab) = f(a) + f(b)$ does not imply that a and b are not immediate neighbors in the ordering. For instance, we could have $a < b < c$, $f(ab) > f(a) + f(b)$, $f(bc) = f(b) + f(c)$, and $f(abc) > f(ab) + f(c)$ (i.e., the weight of abc in $H_I(f)$ is greater than zero).

The following theorem shows that poly-time elicitation is feasible in scenarios a) and b). Determining elicitation complexity under the scenario c) remains open.

Theorem 2 *If $f \in \mathbf{INT}$, then:*

- *Scenario a): it can be elicited in polynomial time asking $\frac{m(m+1)}{2}$ value queries;*
- *Scenario b): it can be elicited in polynomial time asking at most $m^2 - m + 1$ value queries.*

3.6 Tree valuation functions

A natural way to extend the **INT** class is to consider those valuation functions in which the relationships between the objects on sale have a tree structure. Unfortunately, it turns out that the valuation functions that belong to this class, which we denote **TREE**, are not poly-query elicitable even if the structure of the tree is known to the elicitor.

Theorem 3 *There exists a valuation function $f \in \mathbf{TREE}$ that can be learned correctly only asking at least $2^{m/2}$ value queries, even if the elicitor knows the structure of the tree.*

However, if we restrict the super-additive valuations to be only on subtrees of the tree T that describes the item relationships, rather than on arbitrary connected subgraphs of T, then polynomial time elicitation with value queries is possible (given that T itself can be learned in polytime using value queries).

Theorem 4 *Assume that the valuation function $f \in$ **TREE** is such that super-additive valuations are only displayed between objects that form a subtree of T, and assume that the elicitor can learn T asking a polynomial number of value queries. Then, f can be elicited asking a polynomial number of value queries.*

4 Generalized preference elicitation

In the previous section we have considered several classes of valuation functions, proving that most of them are in PTE. However, the definition of PTE (and of PQE) assumes that the elicitor has access to a description of the class of the valuation to elicit; in other words, *the elicitor a priori knows the class to which the valuation function belongs.* In this section, we analyze preference elicitation under a more general framework, in which the elicitor has some uncertainty about the actual class to which the valuation to elicit belongs.

We start by showing that the family of poly-query elicitable classes of valuations is closed under union.

Theorem 5 *Let $\mathbf{C_1}$ and $\mathbf{C_2}$ be two classes of poly-query elicitable valuations, and assume that $p_1(m)$ (resp., $p_2(m)$) is a polynomial such that any valuation in $\mathbf{C_1}$ (resp., $\mathbf{C_2}$) can be elicited asking at most $p_1(m)$ (resp., $p_2(m)$) queries. Then, any valuation in $\mathbf{C_1} \bigcup \mathbf{C_2}$ can be elicited asking at most $p_1(m)+p_2(m)+1$ queries.*

In the following theorem, we prove that the bound on the number of queries needed to elicit a function in $\mathbf{C_1} \bigcup \mathbf{C_2}$ stated in Theorem 5 is tight.

Theorem 6 *There exist families of valuation functions $\mathbf{C_1}, \mathbf{C_2}$ such that either $\mathbf{C_i}$ can be elicited asking at most $m-1$ queries, but $\mathbf{C_1} \cup \mathbf{C_2}$ cannot be elicited asking less than $2m-1 = 2(m-1)+1$ queries (in the worst case).*

Theorem 5 shows that, as far as communication complexity is concerned, efficient elicitation can be implemented under a very general scenario: if the only information available to the elicitor is that $f \in \mathbf{C_1} \bigcup \cdots \bigcup \mathbf{C_{q(m)}}$, where the $\mathbf{C_i}$s are in PQE and $q(m)$ is an arbitrary polynomial, then elicitation can be done with polynomially many queries. This is a notable improvement over traditional elicitation techniques, in which it is assumed that the elicitor knows exactly the class to which the function to elicit belongs.

Although interesting, Theorem 5 leaves open the question of the *computational* complexity of the elicitation process. In fact, the general elicitation algorithm $A_{1 \bigcup 2}$ used in the proof of the theorem (see the full version of the paper [19]) has running time which is super-polynomial in m. So, a natural question to ask is the following: let $\mathbf{C_1}$ and $\mathbf{C_2}$ be *poly-time* elicitable classes of valuations; Is the $\mathbf{C_1} \bigcup \mathbf{C_2}$ class elicitable in polynomial *time*?

Even if we do not know the answer to this question in general, in the following we show that, at least for many of the classes considered in this paper, the answer is *yes.* In particular, we present a polynomial time algorithm that elicits correctly any function $f \in \mathbf{RO_{+M}} \bigcup \mathbf{Tool_{-t}} \bigcup \mathbf{Tool_t} \bigcup \mathbf{G_2} \bigcup \mathbf{INT}$. The algorithm is called GenPolyLearn, and is based on a set of theorems which show that, given any $f \in \mathbf{C_1} \bigcup \mathbf{C_2}$, where $\mathbf{C_1}, \mathbf{C_2}$ are any two of the classes listed above, f can be learned correctly with a low-order polynomial bound on the runtime (see [19]).

The algorithm, which is reported in Figure 1, is very simple: initially, the hypothesis set Hp contains all the five classes. After asking the value of any singleton, GenPolyLearn asks the value of any two-item bundles and, based on the corresponding weights on $H_I(f)$, discards some of the hypotheses. When the hypotheses set contains at most two classes, the algorithm continues preference elicitation accordingly. In case Hp contains more than two classes after all the two-item bundles have been elicited, one more value query (on the grand bundle) is sufficient for the elicitor to resolve uncertainty, reducing the size of the hypotheses set to at most two. The following theorem shows the correctness of GenPolyLearn, and gives a bound on its runtime.

Theorem 7 *Algorithm* GenPolyLearn *learns correctly in polynomial time any valuation function in* $\mathbf{RO_{+M}} \bigcup \mathbf{Tool_{-t}} \bigcup \mathbf{Tool_t} \bigcup \mathbf{G_2} \bigcup \mathbf{INT}$ *asking at most* $O(m(m+t))$ *value queries.*

From the bidders' side, a positive feature of GenPolyLearn is that it asks relatively easy to answer queries: valuation of singletons, two-item bundles, and the grand bundle. (In many cases, the overall value of the market considered (e.g., all the spectrum frequencies in the US) is publicly available information.)

5 Towards characterizing poly-query elicitation

In the previous sections we have presented several classes of valuation functions that can be elicited asking polynomially many queries, and we have proved that efficient elicitation can be implemeted in a quite general setting. In this section, we discuss the properties that these classes have in common, thus making a step forward in the characterization of what renders a class of valuations easy to elicit with value queries.

Let $\mathbf{C}$ be a class of valuations, f any valuation in $\mathbf{C}$, and $A_{\mathbf{C}}$ an elicitation algorithm for $\mathbf{C}$[9]. Let $\mathcal{Q}$ be an arbitrary set of value queries, representing the queries asked by $A_{\mathbf{C}}$ at a certain stage of the elicitation process. Given the answers to the queries in $\mathcal{Q}$, which we denote $\mathcal{Q}(f)$ (f is the function to be elicited), and a description of the class $\mathbf{C}$, $A_{\mathbf{C}}$ returns a set of *learned values*

[9] In the following, we assume that the elicitation algorithm is a "smart" algorithm for $\mathbf{C}$, i.e. an algorithm which is able to infer the largest amount of knowledge from the answers to the queries asked so far.

$V_{\mathbf{C}}(\mathcal{Q}(f))$. This set obviously contains any S such that $q(S) \in \mathcal{Q}$; furthermore, it may contain the value of other bundles (the *inferred values*), which are inferred given the description of $\mathbf{C}$ and the answers to the queries in $\mathcal{Q}$. The elicitation process ends when $V_{\mathbf{C}}(\mathcal{Q}(f)) = 2^I$.

Definition 6 (Inferability) *Let S be an arbitrary bundle, and let f be any function in $\mathbf{C}$. The f-inferability of S w.r.t. $\mathbf{C}$ is defined as:*

$$IN_{f,\mathbf{C}}(S) = \min\left\{|\mathcal{Q}| \text{ s.t. } (q(S) \notin \mathcal{Q}) \text{ and } (S \in V_{\mathbf{C}}(\mathcal{Q}(f)))\right\} .$$

If the value of S can be learned only by asking $q(S)$, we set $IN_{f,\mathbf{C}}(S) = 2^m - 1$. The inferability of S w.r.t. to $\mathbf{C}$ is defined as:

$$IN_{\mathbf{C}}(S) = \max_{f \in \mathbf{C}} IN_{f,\mathbf{C}}(S) .$$

Intuitively, the inferability[10] of a bundle measures how easy it is for an elicitation algorithm to learn the value of S without explicitly asking it.

Definition 7 (Polynomially-inferable bundle) *A bundle S is said to be polynomially-inferable (inferable for short) w.r.t. $\mathbf{C}$ if $IN_{\mathbf{C}}(S) = p(m)$, for some polynomial $p(m)$.*

Definition 8 (Polynomially non-inferable bundle) *A bundle S is said to be polynomially non-inferable (non-inferable for short) w.r.t. $\mathbf{C}$ if $IN_{\mathbf{C}}(S)$ is super-polynomial in m.*

Definition 9 (Strongly polynomially non-inferable bundle) *A bundle S is said to be strongly polynomially non-inferable (strongly non-inferable for short) with respect to class $\mathbf{C}$ if $\forall f \in \mathbf{C}$, $IN_{f,\mathbf{C}}(S)$ is super-polynomial in m.*

Note the difference between poly and strongly poly non-inferable bundle: in the former case, there exists a function f in $\mathbf{C}$ such that, on input f, the value of S can be learned with polynomially many queries only by asking $q(S)$; in the latter case, this property holds *for all* the valuations in $\mathbf{C}$.

Definition 10 (Non-inferable set) *Given a class of valuations $\mathbf{C}$, the non-inferable set of $\mathbf{C}$, denoted $NI_{\mathbf{C}}$, is the set of all bundles in 2^I that are non-inferable w.r.t. $\mathbf{C}$.*

Definition 11 (Strongly non-inferable set) *Given a class of valuations $\mathbf{C}$, the non-inferable set of $\mathbf{C}$, denoted $SNI_{\mathbf{C}}$, is the set of all bundles in 2^I that are strongly non-inferable w.r.t. $\mathbf{C}$.*

[10] When clear from the context, we simply speak of inferability, instead of inferability w.r.t. $\mathbf{C}$.

Clearly, we have $SNI_{\mathbf{C}} \subseteq NI_{\mathbf{C}}$. The following theorem shows that for some class of valuations **C** the inclusion is strict. Actually, the gap between the size of $SNI_{\mathbf{C}}$ and that of $NI_{\mathbf{C}}$ can be super-polynomial in m.

The theorem uses a class of valuations introduced by Angluin [1] in the related context of concept learning. The class, which we call **RDNF** (RestrictedDNF) since it is a subclass of DNF formulas, is defined as follows. There are $m = 2k$ items, for some $k > 0$. The items are arbitrarily partitioned into k pairs, which we denote S_i, with $i = 1, \ldots, k$. We also define a bundle $\bar{S}$ of cardinality k such that $\forall i, |S_i \bigcap \bar{S}| = 1$. In other words, $\bar{S}$ is an arbitrary bundle obtained by taking exactly one element from each of the pairs. We call the S_is and the bundle $\bar{S}$ the minterms of the valuation function f. The valuations in **RDNF** are defined as follows: $f(S) = 1$ if S contains one of the minterms; $f(S) = 0$ otherwise.

Theorem 8 *We have $|SNI_{\mathbf{RDNF}}| = 0$, while $|NI_{\mathbf{RDNF}}|$ is super-polynomial in m.*

Proof. We first prove that $|SNI_{\mathbf{RDNF}}| = 0$. Let f be any function in **RDNF**, and let $S_1, \ldots, S_k, \bar{S}$ be its minterms. Let S be an arbitrary bundle, and assume that S is not a minterm. Then, the value of S can be inferred given the answers to the queries $Q' = \{q(S_1), \ldots, q(S_k), q(\bar{S})\}$, which are polynomially many. Thus, S is not in $SNI_{\mathbf{RDNF}}$. Since for any bundle S there exists a function f in **RDNF** such that S is not one of the minterms of f, we have that $SNI_{\mathbf{RDNF}}$ is empty. Let us now consider $NI_{\mathbf{RDNF}}$. Let S be an arbitrary bundle of cardinality k, and let f be a function in **RDNF**. If S is one of the minterms of f (i.e., $S = \bar{S}$) the only possibility for the elicitor to infer its value is by asking the value of *all* the other bundles of cardinality k (there are super-polynomially many such bundles). In fact, queries on bundles of cardinality $< k$ of $\geq k + 1$ give no information on the identity of $\bar{S}$. So, $\bar{S}$ is in $NI_{\mathbf{RDNF}}$. Since for any bundle S of cardinality k there exists a function f in **RDNF** such that S is a minterm of f, we have that $NI_{\mathbf{RDNF}}$ contains super-polynomially many bundles.

The following theorem shows that whether a certain class **C** is in PQE depends to a certain extent on the size of $SNI_{\mathbf{C}}$.

Theorem 9 *Let **C** be an arbitrary class of valuations. If the size of $SNI_{\mathbf{C}}$ is super-polynomial in m, then $\mathbf{C} \notin PQE$.*

Theorem 9 states that a necessary condition for a class of valuations **C** to be easy to elicit is that its strongly non-inferable set has polynomial size. Is this condition also sufficient? The following theorem, whose proof follows immediately by the fact that the **RDNF** class is hard to elicit with value queries [1], gives a negative answer to this question, showing that even classes **C** with an empty strongly non-inferable set may be hard to elicit.

Theorem 10 *The condition $|SNI_{\mathbf{C}}| = p(m)$ for some polynomial $p(m)$ is not sufficient for making **C** easy to elicit with value queries. In particular, we have that $|SNI_{\mathbf{RDNF}}| = 0$, and $\mathbf{RDNF} \notin PQE$.*

Theorem 10 shows that the size of the strongly non-inferable set alone is not sufficient to characterize classes of valuations which are easy to elicit. Curiously, the size of the non-inferable set of **RDNF** is super-polynomial in m. Thus, the following question remains open: "Does there exist a class of valuations $\mathbf{C}$ such that $|NI_{\mathbf{C}}| = p(m)$ for some polynomial $p(m)$ and $\mathbf{C} \notin PQE$?" or, equivalently, "Is the condition $|NI_{\mathbf{C}}| = p(m)$ for some polynomial $p(m)$ sufficient for making $\mathbf{C}$ poly-query elicitable?"

Furthermore, Theorem 10 suggests the definition of another notion of poly-query elicitation, which we call "non-deterministic poly-query elicitation" and denote with NPQE. Let us consider the **RDNF** class used in the proof of Theorem 8. In a certain sense, this class seems easier to elicit than a class $\mathbf{C}$ with $|SNI_{\mathbf{C}}|$ superpolynomial in m. In case of the class $\mathbf{C}$, any set of polynomially many queries is not sufficient to learn the function (no "poly-query certificate" exists). Conversely, in case of **RDNF** such "poly-query certificate" exists for any $f \in$ **RDNF** (it is the set $\mathcal{Q}'$ as defined in the proof of Theorem 8); what makes elicitation hard in this case is the fact that this certificate is "hard to guess". So, the **RDNF** class is easy to elicit if non-deterministic elicitation is allowed. The following definition captures this concept:

Definition 12 (NPQE) *A class of valuations $\mathbf{C}$ is said to be poly-query non-deterministic (fully) elicitable if there exists a nondeterministic elicitation algorithm which, given as input a description of $\mathbf{C}$, and by asking value queries only, learns any valuation $f \in \mathbf{C}$ asking at most $p(m)$ queries in at least one of the nondeterministic computations, for some polynomial $p(m)$. NPQE is the set of all classes $\mathbf{C}$ that are poly-query nondeterministic elicitable.*

It turns out that non-deterministic poly-query elicitation can be characterized using a notion introduced in [10], which we adapt here to the framework of preference elicitation.

Definition 13 (Teaching dimension) *Let $\mathbf{C}$ be a class of valuations, and let f be an arbitrary function in $\mathbf{C}$. A teaching set for f w.r.t. $\mathbf{C}$ is a set of queries $\mathcal{Q}$ such that $V_{\mathbf{C}}(\mathcal{Q}(f)) = 2^I$. The teaching dimension of $\mathbf{C}$ is defined as*

$$TD(\mathbf{C}) = \max_{f \in \mathbf{C}} \min \left\{ |\mathcal{Q}| \text{ s.t. } (\mathcal{Q} \subseteq 2^{2^I}) \text{ and } (\mathcal{Q} \text{ is a teaching set for } f) \right\} .$$

Theorem 11 *Let $\mathbf{C}$ be an arbitrary class of valuations. $\mathbf{C} \in NPEQ$ if and only if $TD(\mathbf{C}) = p(m)$ for some polynomial $p(m)$.*

The following results is straightforward by observing that **RDNF** is in NPQE (it has $O(m)$ teaching dimension) but not in PQE:

Proposition 2 *$PQE \subset NPQE$.*

References

1. D. Angluin, "Queries and Concept Learning", *Machine Learning*, Vol. 2, pp. 319–342, 1988.

2. A. Blum, J. Jackson, T. Sandholm, M. Zinkevic, "Preference Elicitation and Query Learning", *in Proc. Conference on Computational Learning Theory (COLT)*, 2003.
3. L. Blumrosen, N. Nisan, "Auctions with Severely Bounded Communication", *in Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 406–415, 2002.
4. A. Bonaccorsi, B. Codenotti, N. Dimitri, M. Leoncini, G. Resta, P. Santi, "Generating Realistic Data Sets for Combinatorial Auctions", *Proc. IEEE Conf. on Electronic Commerce (CEC)*, pp. 331–338, 2003.
5. E.H. Clarke, "Multipart Pricing of Public Goods", *Public Choice*, Vol. 11, pp. 17–33, 1971.
6. W. Conen, T. Sandholm, "Preference Elicitation in Combinatorial Auctions", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 256–259, 2001. A more detailed description of the algorithmic aspects appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.
7. W. Conen, T. Sandholm, "Partial-Revelation VCG Mechanisms for Combinatorial Auctions", *Proc. National Conference on Artificial Intelligence (AAAI)*, pp. 367–372, 2002.
8. V. Conitzer, T. Sandholm, P. Santi, "On K-wise Dependent Valuations in Combinatorial Auctions", *internet draft.*
9. S. de Vries, R. Vohra, "Combinatorial Auctions: a Survey", *INFORMS J. of Computing*, 2003.
10. S. Goldman, M.J. Kearns, "On the Complexity of Teaching", *Journal of Computer and System Sciences*, Vol. 50, n. 1, pp. 20–31, 1995.
11. T. Groves, "Incentive in Teams", *Econometrica*, Vol. 41, pp. 617–631, 1973.
12. B. Hudson, T. Sandholm, "Effectiveness of Query Types and Policies for Preference Elicitation in Combinatorial Auctions", *International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, 2004.
13. D. Lehmann, L. Ita O'Callaghan, Y. Shoham, "Truth Revelation in Approximately Efficient Combinatorial Auctions", *Journal of the ACM*, Vol.49, n.5, pp. 577–602, 2002.
14. J. MacKie-Mason, H.R. Varian, "Generalized Vickrey Auctions", *working paper*, Univ. of Michigan, 1994.
15. N. Nisan, I. Segal, "The Communication Requirements of Efficient Allocations and Supporting Lindhal Prices", *internet draft*, version March 2003.
16. M.H. Rothkopf, A. Pekec, R.H. Harstad, "Computationally Managable Combinatorial Auctions", *Management Science*, Vol. 44, n. 8, pp. 1131–1147, 1998.
17. T. Sandholm, S. Suri, "BOB: Improved Winner Determination in Combinatorial Auctions and Generalizations", *Artificial Intelligence*, Vol. 145, pp. 33–58, 2003.
18. T. Sandholm, "Algorithm for Optimal Winner Determination in Combinatorial Auctions", *Artificial Intelligence*, Vol. 135, pp. 1–54, 2002.
19. P. Santi, V. Conitzer, T. Sandholm, "Towardsa a Characterization of Polynomial Preference Elicitation with Value Queries in Combinatorial Auctions", *internet draft*, available at *http://www.imc.pi.cnr.it/∼santi.*
20. W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders", *Journal of Finance*, Vol. 16, pp. 8–37, 1961.
21. M. Zinkevich, A. Blum, T. Sandholm, "On Polynomial-Time Preference Elicitation with Value Queries", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 176–185, 2003.
22. E. Zurel, N. Nisan, "An Efficient Approximate Allocation Algorithm for Combinatorial Auctions", *Proc. 3rd ACM Conference on Electronic Commerce (EC)*, pp. 125–136, 2001.

Algorithm GENPOLYLEARN:

```
0. Hp={$\mathbf{RO_{+M}}$,$\mathbf{G_2}$,$\mathbf{Tool_t}$,$\mathbf{Tool_{-t}}$,INT}
1. build the first level of $H_I(f)$ asking the value of singletons

2. build the second level of $H_I(f)$ asking the value of two-items bundles in arbitrary order
3. let $w(ab)$ the computed weight for bundle $ab$
4. repeat
5.     if $w(ab) < 0$ then
6.        remove $\mathbf{Tool_t}$ and INT from Hp
7.        if $w(ab) \neq -\min\{f(a), f(b)\}$ then remove $\mathbf{RO_{+M}}$ from Hp
8.     if $w(ab) > 0$ then
9.           remove $\mathbf{RO_{+M}}$ and $\mathbf{Tool_{-t}}$ from Hp
10.          if $w(ab)$ is not compatible with the linear order discovered so far then
11.             remove INT from Hp
12. until $|\mathrm{Hp}| \leq 2$ or all the $w(ab)$ have been considered
13. if $|\mathrm{Hp}| \leq 2$ then continue elicitation as described in theorems 6–15 of [19].

otherwise:
14. case 1: all the $w(ab)$ weights are $\geq 0$ and compatible with the linear order, and at least one weight is positive
15.     ask the value of the grand bundle $I$
16.     if $f(I) = \sum_{S \subseteq I, |S| \leq 2} w(S)$ then
17.         remove $\mathbf{Tool_t}$ from Hp
18.         continue elicitation as in the proof of Th. 9 of [19]
19.     else
20.         remove $\mathbf{G_2}$ from Hp
21.         continue elicitation as in the proof of Th. 10 of [19]
22. case 2: all the $w(ab)$ weights are $\leq 0$, at least one weight is negative, and $\mathbf{RO_{+M}} \in$Hp
23.     ask the value of the grand bundle $I$
24.     if $f(I) \neq \sum_{S \subseteq I, |S| \leq 2} w(S)$ then
25.         remove $\mathbf{G_2}$ from Hp
26.         continue elicitation as in the proof of Th. 15 of [19]
27.     else
28.         remove $\mathbf{Tool_{-t}}$ from Hp
29.         continue elicitation as in the proof of Th. 6 of [19]
30. case 3: $w(ab) = 0$ for all $ab$
31.     ask the value of the grand bundle $I$
32.     if $f(I) < \sum_{a \in I} f(a)$ then
33.         remove INT, $\mathbf{Tool_t}$, $\mathbf{G_2}$, $\mathbf{RO_{+M}}$ from Hp
34.         $f \in \mathbf{Tool_{-t}}$; continue elicitation accordingly
35.     else
36.         remove $\mathbf{Tool_{-t}}$, $\mathbf{G_2}$, $\mathbf{RO_{+M}}$ from Hp
37.         proceed as in the proof of Th. 10 of [19]
```

Fig. 1. Algorithm for learning correctly any valuation function in $\mathbf{RO_{+M}} \bigcup \mathbf{Tool_{-t}} \bigcup \mathbf{Tool_t} \bigcup \mathbf{G_2} \bigcup \mathbf{INT}$ asking a polynomial number of value queries.