Towards a Characterization of Polynomial Preference Elicitation with Value Queries in Combinatorial Auctions*

Paolo Santi† Vincent Conitzer‡ Tuomas Sandholm‡

September 17, 2004

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 submitted by the bidders, elicits the bidders' preferences (or valuations) by asking value queries. It is known that in the most 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 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 submit bids on bundle of items, and thus may easily express complementarities (i.e., the bidder values multiple items together more than the sum of the

*A short, early version of this paper appeared at the 17th Annual Conference on Learning Theory (COLT 04), pp. 1–16. This work is supported in part by NSF under CAREER Award IRI-9703122, Grant IIS-9800994, ITR IIS-0081246, and ITR IIS-0121678.

†Istituto di Informatica e Telematica, Pisa, 56124, Italy, Email: paolo.santi@iit.cnr.it. This work was done when the author was visiting the Dept. of Computer Science, Carnegie Mellon University.

‡Dept. of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue Pittsburgh, PA 15213. Emails: {conitzer, sandholm}@cs.cmu.edu

valuations of the individual items), and substitutabilities (i.e., multiple items together are worth less than the sum of the valuations of the individual items) between the objects on sale[1]. CAs can be used, 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 [18], it is shown that solving the winner determination problem is NP-hard; even worse, finding a $n^{1-\epsilon}$-approximation (here, n is the number of bids) to the optimal solution is NP-hard [20]. Despite these hardness 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, 15, 19, 20, 23]. 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, 13]. Unfortunately, a recent result by Nisan and Segal [17] 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, 14, 17, 22]. 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.

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

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

should be contrasted with the other definition of preference elicitation, in which 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, 21] can be computed without further message exchange. Since VCG payments prevent strategic bidding behavior [16], 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 only ask the bidders *value queries* (what is the value of a particular bundle?). 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 i-th most valuable bundle?) queries.

Full preference elicitation with value queries has been investigated in a few recent papers. In [22], 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 (we recall that sub- and super-additivity are the same as substitutability and complementarities, respectively), 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[2] (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 val-

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

uations 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. Thanks to this property, the elicitor does not need to know exactly the class to which valuation functions belong. We remark that the union property above is valid only for elicitation with value queries. For instance, assume elicitation is done with rank queries, and that we have two classes of "easy to elicit with rank queries" valuations $\mathbf{C_1}$ and $\mathbf{C_2}$. Assume there exist $f_1 \in \mathbf{C_1}$ and $f_2 \in \mathbf{C_2}$ such that f_1 and f_2 have the same exact ranking of items (but they give different values to the bundles). In this case, if only rank queries can be used the elicitor has no way of figuring out which one of the two valuations is the right one.

• The algorithm used to elicit valuations in $\mathbf{C_1} \bigcup \mathbf{C_2}$ might have super-polynomial running time (even though it asks only polynomially many queries). Indeed, we show that there exist classes $\mathbf{C_1}, \mathbf{C_2}$ such that eliciting a function in $\mathbf{C_1} \bigcup \mathbf{C_2}$ can be done asking polynomially many queries, but identifying the queries to ask is NP-complete. Despite this hardness result, we present an efficient 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 an 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 [17], 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 denote 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[3], and weight $f(a)$ to any singleton a. Let us now consider a node at level 2, say node ab[4]. 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[5], 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 to 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 $H_I(f)$

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

[4] 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.

[5] 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.

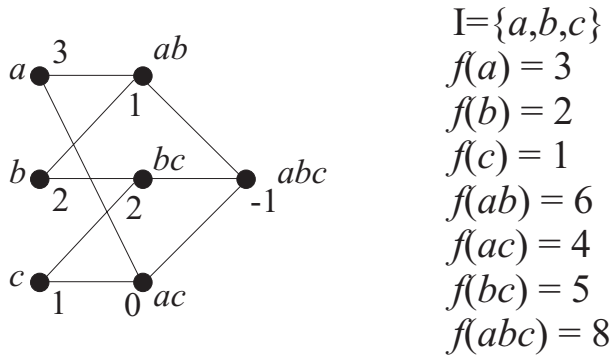

Figure 1: Example of valuation function, and the corresponding hypercube representation. In the hypercube representation, the node corresponding to the empty set has weight 0, and it is not shown.

such that $S' \subseteq S$. These are the only weights contained in the sub-hypercube of $H_I(f)$ "rooted" at S. An example of valuation function and its hypercube representation are reported in Figure 1.

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

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.

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 [22]. 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 [22], 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 $i > k$.*

K-wise dependent valuations can be represented using the k-wise dependency graph, which makes it explicit the dependencies between item valuations. For instance, in case of 2-wise dependent valuations, the 2-wise dependency graph, denoted G_2, is built as follows:

- let there be a node for every item;
- label node a with $f(a)$;[6]
- if a and b are super- or sub-additive, put an (undirected) edge (a, b) in the graph, and label the edge with $f(ab) - (f(a) + f(b))$.

If $f \in \mathbf{G_2}$, the corresponding graph G_2 can be used to calculate the valuation of any possible bundle S as follows: consider the subgraph G^S of G_2 induced by node set S; sum up all the node and edge labels in G^S. An example of 2-wise dependency graph, and the corresponding valuation function, is reported in Figure 2.

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.

[6] Slightly abusing the notation, we use a to denote both the item and the corresponding node in the graph.

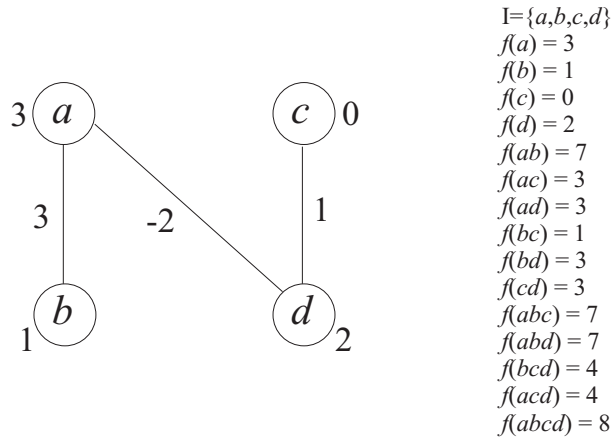

Figure 2: Example of 2-wise dependency graph, and the corresponding valuation function.

3.3 The $\mathbf{Tool_t}$ class

The class of ToolboxDNF formulas, which we denote $\mathbf{Tool_t}$, has been introduced in [22], 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 that 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[7], and can be elicited in polynomial time asking $O(mt)$ value queries [22].

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

This class of valuation functions is a variation of the ToolboxDNF class introduced in [22]. 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

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

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

Proof. The proof is an easy adaptation of the proof of Theorem 8 of [22]. First, we ask the value of all the singletons, thus finding all the minterms of f of cardinality 1. Then, we ask the value of the grand bundle, and we compare its value with the linear valuation f_l, which assigns the value $f_l(S) = \sum_{a \in S} f(a)$ to any bundle. If $f(I) = f_l(I)$, then there are no other minterms for f, and we are done. Otherwise, we repeatedly remove elements from the grand bundle, until we find a minimal set S_1 such that $f(S_1) < f_l(S_1)$, and $f(S_1 - \{a\}) = f_l(S_1 - \{a\})$ for any $a \in S_1$. Bundle S_1, which can be discovered asking at most m value queries, is our next minterm. Preference elicitation is then continued re-defining f_l in order to account for the new minterm, as described in [22]. □

3.5 Interval valuation functions

The class of interval valuations is inspired by the notion of interval bids [18, 19], 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 substitutability-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 $f(a) = 10$, $f(b) = 5$, $f(c) = 3$, $f(ab) = 17$, $f(bc) = 10$, $f(ac) = f(a) + f(c) = 13$ (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 $f \in$ **INT**, then:*

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

Proof. We distinguish the two scenarios:

a) In this case f can be elicited asking m queries for the singletons, $m-1$ queries for the interval two-item bundles, $m-2$ queries for the interval three-item bundles, and so on; i.e., $\frac{m(m+1)}{2}$ queries in total. Since these are the only non-additive valuations in f, these queries are sufficient for the elicitor to learn $H_I(f)$ (hence, f) correctly: the remaining weights in $H_I(f)$ are 0.

b) In this case, the elicitor in the worst case must ask the value of every singleton and two-item bundle in order to identify the item linear order, which can be uniquely identified given the $m-1$ non-zero weights on the second level nodes of $H_I(f)$. Once the linear order is defined, elicitation proceeds as in the previous case. In this case, the total number of queries asked in the worst case is $m + \frac{m(m-1)}{2} + (m-2) + \cdots + 1 = m^2 - m + 1$.

□

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.

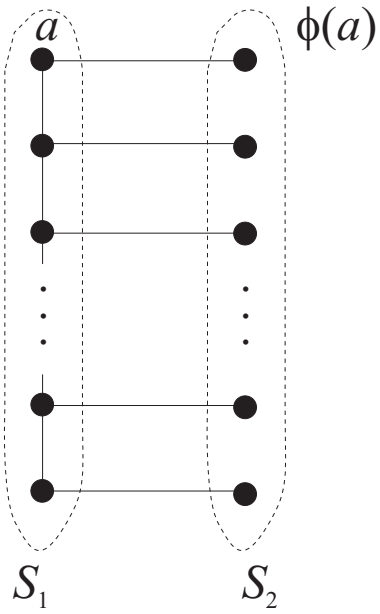

Figure 3: The tree T used in the proof of Theorem 3.

Theorem 3. *There exists a valuation function $f \in$ **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.*

Proof. For simplicity, assume m is even. Let us divide the set of items I into two subsets S_1, S_2 of cardinality $m/2$, and let ϕ be an arbitrary bijection between S_1 and S_2. Let us order the items in S_1 according to an arbitrary linear order. For every item in S_1 we add an edge to the following item in the linear order. Then, we add an edge between any item $a \in S_1$ and the corresponding item $\phi(a) \in S_2$. It is immediate that the resulting graph is a tree, which we denote T (see Figure 3). Let us consider an arbitrary valuation function f such that the bundle relationships between items are described by T. In order to learn f correctly, the elicitor must ask the value of any possible bundle compatible with T, i.e. the value of any bundle that corresponds to a connected subgraph of T. We claim that there are at least $2^{m/2}$ such bundles.
To prove the claim, let us consider the connected subgraph of T induced by the items in S_1, which is a simple path. For every $a \in S_1$, adding $\phi(a)$ to S_1 forms a bundle which is compatible with T. Thus, there exist at least $2^{|S_1|} = 2^{m/2}$ different ways of extending S_1 into a compatible bundle, and the theorem is proved. □

However, if we impose the restriction that the super-additive valuations 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 superadditive 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.*

Proof. The proof is immediate by observing that, once the structure of T is known, the number of possible proper subtrees of T is at most $m-2$; in fact, every node which is not the root of T or a leaf is the root of exactly one subtree, and T contains at least one leaf node. □

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.*

Proof. Since $\mathbf{C_1}$ is poly-query elicitable, there exists a poly-query elicitation algorithm A_1 for $\mathbf{C_1}$. This algorithm, taken as input a description of $\mathbf{C_1}$ (i.e., the elicitor knows that the function to elicit is in $\mathbf{C_1}$), starts asking query q_1^1; given the answer to this query, decides the next query to ask q_2^1, and so on, until elicitation is complete. Since $\mathbf{C_1}$ is poly-query elicitable, this process is guaranteed to end after at most $p_1(m)$ queries have been asked. A similar algorithm A_2, which asks at most $p_2(m)$ queries, exists for class $\mathbf{C_2}$ also.
Given algorithms A_1 and A_2, the algorithm $A_{1 \bigcup 2}$ to elicit any function f in $\mathbf{C_1} \bigcup \mathbf{C_2}$ works as follows. The elicitor has two working hypotheses: $\mathcal{H}_1 = \{f$ is in $\mathbf{C_1}\}$, and $\mathcal{H}_2 = \{f$ is in $\mathbf{C_2}\}$. The goal of the elicitor is to ask queries in order to resolve the uncertainty, showing that, given the answers to the queries asked so far, only one of the two hypotheses holds. Once uncertainty is resolved, elicitation continues using the A_i algorithm correspondent to the valid hypothesis. If uncertainty is never resolved, the elicitor can conclude that

$f \in \mathbf{C_1} \bigcap \mathbf{C_2}$, and elicitation is continued according to either A_1 or A_2 (they are both correct elicitation algorithms for f).
W.l.o.g., assume the elicitor starts asking query q_1^1. In other words, the elicitor assumes that $\mathcal{H}_1$ holds. Given the answer to q_1^1, the elicitor checks whether it is compatible with $\mathcal{H}_1$. If not, it concludes that $\mathcal{H}_2$ holds, and continues elicitation according to A_2. Otherwise, the elicitor checks whether the answer to q_1^1 is compatible with $\mathcal{H}_2$. If not, it concludes that $\mathcal{H}_1$ holds, and continues elicitation according to A_1. If the answer to the first query is not sufficient to resolve uncertainty, the elicitor ask query q_2^1, checks for compatibility with the hypotheses, and so on. In the worst case, all the answers to the queries $q_1^1,\ q_2^1, \ldots, q_p^1$ are compatible with $\mathcal{H}_1$ (and with $\mathcal{H}_2$), so the elicitation process based on $\mathcal{H}_1$ stops after at most polynomially many queries have been asked. At this stage, the elicitor is able to build a hypothetical learned function f_1 based on the answers to queries $q_1^1,\ q_2^1, \ldots, q_p^1$. Now, the elicitor starts elicitation assuming $\mathcal{H}_2$ holds: it asks query q_1^2 (if not already asked), checks for compatibility with $\mathcal{H}_1$ and $\mathcal{H}_2$, and so on. This process stops when:

- uncertainty is resolved. In this case, if $\mathcal{H}_1$ holds, the elicitor can correctly conclude that $f = f_1$; otherwise, it continues elicitation according to A_2.
- if uncertainty is not resolved, after asking at most $q \leq p_2(m)$ queries the elicitor can build a second hypothetical learned function f_2.

In order to complete elicitation, the elicitor compares the value of f_1 and f_2 on all possible bundles. If there exists any bundle $\bar{S}$ such that $f_1(\bar{S}) \neq f_2(\bar{S})$, then the elicitor asks query $q(\bar{S})$, and concludes elicitation accordingly ($f = f_1$ if $f(\bar{S}) = f_1(\bar{S})$, $f = f_2$ otherwise); otherwise, it must be $f = f_1 = f_2$ (i.e., $f \in \mathbf{C_1} \bigcap \mathbf{C_2}$).
To end the proof of the theorem, it is sufficient to observe that in the worst case $A_{1 \bigcup 2}$ requires $p_1(m) + p_2(m) + 1$ queries to complete elicitation. □

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).*

Proof. Consider the class $\mathbf{C_1}$ defined as follows: for every item $a \in I$, there is a function $f_1^a \in \mathbf{C_1}$ given by $f_1^a(\emptyset) = 0$, $f_1^a(\{a\}) = 0$, $f_1^a(I) = 2$, and $f_1^a(S) = 1$ for every other bundle S. We observe that the only uncertainty is in which singleton bundle has value 0. Thus, any function in $\mathbf{C_1}$ is elicitable by asking at most $m-1$ value queries on singleton bundles (if none of these queries return 0, the query on the last singleton must return 0).
Now consider the class $\mathbf{C_2}$ defined as follows: for every item $a \in I$, there is a function $f_2^a \in \mathbf{C_2}$ given by $f_2^a(\emptyset) = 0$, $f_2^a(I - \{a\}) = 2$, $f_2^a(I) = 2$, and $f_2^a(S) = 1$ for every other bundle S. We observe that the only uncertainty is in which bundle of $m-1$ items has value 2. Thus, any function in $\mathbf{C_2}$ is elicitable

by asking at most $m-1$ value queries on bundles of size $m-1$ (if none of these queries return 2, the query on the last bundle of size $m-1$ must return 2). Now consider eliciting functions from $\mathbf{C_1} \cup \mathbf{C_2}$. The only queries where it is not certain what the value returned will be are the $2m$ queries on bundles of size 1 or $m-1$. Exactly one such query will return a value different from 1; but we will not know which one until we have asked that query, or we have asked all the other queries on bundles of size 1 or $m-1$. It follows that we need at least $2m-1$ 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 has a worst-case 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*?

In the following, we show that the answer to this question is negative. In order to prove the result, we define two classes of valuation functions:

Definition 6 ($\mathbf{G_2^U}$). *A valuation function f in $\mathbf{G_2^U}$ is defined by a valuation function $f' \in \mathbf{G_2}$ (with only nonnegative weights on the edges), and an upper bound u on the value of the bundles. The value of a bundle S is $f(S) = \min\{f'(S), u\}$. That is, f is the same as f' except its values cannot exceed u.*

A function f in $\mathbf{G_2^U}$ is easy to elicit: ask all the singleton and two-item bundles to get f', as well as the grand bundle to get u.

Definition 7 ($\mathbf{G_2^{UH}}$). *A valuation function f in $\mathbf{G_2^{UH}}$ is also defined by a valuation function $f' \in \mathbf{G_2}$ (with only nonnegative weights on the edges), and an upper bound u on the value of the bundles. Except, for this class, there is the additional constraint that no more than half the value of a bundle can come from the edge weights. That is, $f(S) = \min\{f'(S), 2\sum_{a \in S} f'(a), u\}$. (We also require that for f', the value placed on an edge of the graph does not exceed the sum of its neighboring vertices, that is, $f'(a,b) \leq 2(f'(a) + f'(b))$.)*

Again, a function f in $\mathbf{G_2^{UH}}$ is easy to elicit: ask all the singleton and pair bundles to get f', as well as the grand bundle to get u.

Theorem 7. *It is coNP-complete to tell if a given function f_1 in $\mathbf{G_2^U}$ and a given function f_2 in $\mathbf{G_2^{UH}}$ (represented by their f' valuations and their u) are identical or not. It follows that it is just as hard to determine a query that would*

distinguish them (because otherwise, we could easily tell if they were identical or not).

Proof. The problem is in coNP because a bundle on which the functions are different is a certificate for their inequality. To show the problem is coNP-hard, we reduce an arbitrary CLIQUE problem instance (given by a graph G, and a number k such that we want to find a set of k vertices with edges between every pair of these vertices) to the following function distinguishing problem. For both f_1 in $\mathbf{G_2^U}$ and f_2 in $\mathbf{G_2^{UH}}$, let f' be defined as follows. Its G_2 graph is the given G from the CLIQUE instance. The weight on every vertex is 1. The weight on every edge is $\frac{k+\epsilon}{\binom{k}{2}}$. Also, for both f_1 in $\mathbf{G_2^U}$ and f_2 in $\mathbf{G_2^{UH}}$, let $u = 2k + \epsilon$. We claim that (if epsilon is sufficiently small) the functions are different if and only if there is a solution to the CLIQUE problem.

First suppose there exists a solution to the CLIQUE problem. Then, the functions differ on the bundle corresponding to the clique of size k: f_1's value on this bundle will be $2k+\epsilon$, but f_2's value will be only $2k$ (because the vertices contribute only k).

Now suppose there is no solution to the CLIQUE problem. Then to show that (if epsilon is sufficiently small) the functions will be the same, we only have to show that the constraint that the total value contributed by the edges cannot exceed the total value contributed by the vertices is never binding. First, on any bundle of size less than k, the total value of the edges will not exceed the total value of the vertices. This is because even if the nodes constitute a clique (of size less than k), the ratio of the total edge value to the total vertex value is strictly less than that same ratio would be for a clique of size k – and for a clique of size k this ratio only barely exceeds 1. Second, on any bundle of size k, this ratio must also be smaller than 1, because there is at least one fewer edge in this bundle than in a clique of size k (because there is no clique of size k). Finally, if the total value of the edges exceeds the total value of the vertices on a bundle of size greater than k, then the upper bound constraint u is the binding constraint for both functions because $2(k+1) > 2k + \epsilon$. □

Despite the hardness result proved above, polynomial time preference elicitation is possible for the union set of several classes of valuations of practical interest. 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 the following 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.

Theorem 8. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{G_2} \bigcup \mathbf{RO_{+M}}$. Then, the elicitor can learn f correctly in polynomial time by asking at most $\frac{m(m+1)}{2} + 1$ value queries.*

Proof. If the f is in $\mathbf{G_2}$, it can be learned correctly by asking the value of every bundle of at most 2 items. Similarly, if f is in $\mathbf{RO_{+M}}$, it can be learned

correctly by asking the value of every bundle of at most 2 items [22]. In the latter case, however, the answers to the queries on two-item bundles cannot have arbitrary values. In fact, we have $f(ab) = f(a) + f(b)$ if the least common ancestor (LCA) of a and b in the read-once formula corresponding to f is a SUM gate, and $f(ab) = \max\{f(a), f(b)\}$ if it is a MAX gate. Thus, if any one of the answers to the queries on two-item bundles does not satisfy $f(ab) = (f(a) + f(b))$ or $(\max\{f(a), f(b)\})$ the elicitor knows that f must be in $\mathbf{G_2}$, and we are done.

Unfortunately, it could be the case that the answers to all the two-item queries are compatible with the hypothesis that $f \in \mathbf{RO_{+M}}$, but actually $f \notin \mathbf{RO_{+M}}$. For instance, suppose $f(a) = 3$, $f(b) = 2$, $f(c) = 8$, $f(ab) = 3$, $f(bc) = 8$ and $f(ac) = 8$; then, the value of abc is 6 if $f \in \mathbf{G_2}$, while it is 8 if $f \in \mathbf{RO_{+M}}$.

However, in this case one single query on a three-items bundle is sufficient for the elicitor to learn f correctly. To identify the query to ask, the elicitor builds the (unique) read-once formula coherent with the values of the singletons and two-items bundles elicited, as described in Lemma 1 of [22] (here, we are assuming that the individual items have nonzero valuation). Let T be the tree corresponding to this formula, where the root of T represents the outcome of the formula. Every non-leaf node in T is either a SUM or MAX operator, and has at least two children. The elicitor scans T starting from the root, identifying the MAX operator(s) of minimum depth. Let $M_1, \ldots, M_c$ be these operators. Assume that at least one of them (say, M_1) has at least three children, denoted C_1, C_2, C_3. If so, it is sufficient to ask the value of any bundle abc such that a, b, and c are leaf nodes contained in the subtrees rooted at C_1, C_2, and C_3, respectively; if $f(abc) = \max\{f(a), f(b), f(c)\}$, then f must be in $\mathbf{RO_{+M}}$ (in fact, $\max\{f(a), f(b), f(c)\}$ is a 3-wise dependency between items, which is excluded in valuations in class $\mathbf{G_2}$), otherwise we must have $f \in \mathbf{G_2}$. Assume now that all the M_i have two children, but there exists at least one MAX gate, say M_1, such that the subtree rooted at M_1 contains at least one MAX gate (excluding M_1). Let $\bar{M}$ denote one such gate. Both M_1 and $\bar{M}$ have at least two children, denoted C_1^1, C_1^2 and $\bar{C}^1$, $\bar{C}^2$, respectively. W.l.o.g., assume $\bar{M}$ is in the subtree rooted at C_1^1. Let us consider any bundle abc such that a is in the subtree rooted at $\bar{C}^1$, b is in the subtree rooted at $\bar{C}^2$, and c is in the subtree rooted at C_1^2. Asking the value of abc, the elicitor can determine to which class the function to be learned belongs: if $f(abc) = \max\{f(a), f(b), f(c)\}$, then $f \in \mathbf{RO_{+M}}$, otherwise $f \in \mathbf{G_2}$. If none of the MAX gates $M_1, \ldots, M_c$ satisfy one of the conditions above, it means that all the M_i have two children, and that there exists only SUM gates in the subtrees rooted at them. If all the subtrees rooted at the M_is have at most two leaves, it means that f can be expressed using both a read-once formula and a G_2 graph; i.e., $f \in \mathbf{G_2} \bigcap \mathbf{RO_{+M}}$. This is because the MAX gate is the only operator that introduces dependencies between item valuations; if all the subtrees rooted at the M_is have at most two leaves, there are at most 2-wise dependencies between items, and the valuation function is in $\mathbf{G_2}$. Thus, f is learned correctly from the initial $\frac{m(m+1)}{2}$ queries, using either the G_2 graph or the read-once formula to calculate the value of f

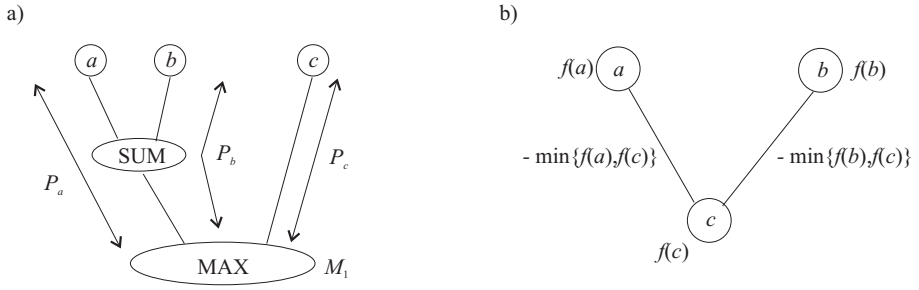

Figure 4: a) Subtree rooted at M_1 b) Corresponding portion of the G_2 graph.

on any other bundle. Assume now that there exists at least one M_i, say M_1, such that the subtree rooted at M_1 has at least three leaves. Let us denote with a, b, and c any three of these leaves. Let us consider the paths P_a, P_b and P_c connecting a, b and c to M_1 in the formula. Since all the gates in the subtree rooted at M_1 are SUM gates, it follows that P_a, P_b and P_c contain only SUM nodes (excluding the starting and ending nodes of the path). Furthermore, since M_1 has two children, two of the three paths, say P_a and P_b, must have a non-empty intersection (excluding the destination node M_1). The resulting subtree structure is depicted in Figure 4.a. If $f \in \mathbf{RO_{+M}}$, we must have $f(abc) = \max\{f(a)+f(b), f(c)\}$. On the other hand, the subgraph induced on G_2 by the node set abc has the structure shown in Figure 4.b, which gives a value equal to $f(a)+f(b)+f(c)-\min\{f(a),f(c)\}-\min\{f(b),f(c)\}$ to bundle abc. It follows that, unless $f(a)+f(b)=f(c)$, asking the value of the bundle abc is sufficient for the elicitor to determine whether f is in $\mathbf{RO_{+M}}$ or in $\mathbf{G_2}$. If $f(a)+f(b)=f(c)$, the read-once formula and the G_2 graph give the same valuation to the bundle abc. It follows that, if no bundle abc as in the construction above such that $f(a)+f(b) \neq f(c)$ exists, then f can be expressed using both a read-once formula and the G_2 graph. Then, $f \in \mathbf{G_2} \bigcap \mathbf{RO_{+M}}$, and the value of f on any bundle can be learned correctly from the initial $\frac{m(m+1)}{2}$ queries, using either the G_2 graph or the read-once formula. □

Theorem 9. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{G_2} \bigcup \mathbf{Tool_t}$, with t polynomial in m. Then, the elicitor can learn f correctly in polynomial time by asking at most $\frac{m(m+1)}{2} + O(tm)$ value queries.*

Proof. The elicitor asks the value of every bundle of at most two items, and uses the answers to these queries to build $H_I(f)$ up to level 2. If $f \in \mathbf{G_2}$, then $w(S) = 0$ for every node in H_I corresponding to set S, with $|S| > 2$. So, the problem is to verify whether there exists at least one node in H_I at level > 2 with nonzero weight. Assume that $f \in \mathbf{Tool_t}$. It is immediate that $w(S) > 0$ if and only if bundle S is one of the minterms of f, and that $w(S) = 0$ otherwise. If all the minterms of f are composed of at most two items, then we

have $f \in \mathbf{G_2} \bigcap \mathbf{Tool_t}$, and we are done. Otherwise, there must exist at least one minterm S with at least three items, i.e. $w(S) > 0$ for some S with $|S| > 2$. Since the weights in $H_I(f)$ cannot be negative if $f \in \mathbf{Tool_t}$, the uncertainty can be solved by asking the valuation of the grand bundle I: if $f(I) = \sum_{S:|S|\leq 2} w(S)$ then $f \in \mathbf{G_2}$ and we are done; otherwise, it must be $f \in \mathbf{Tool_t}$, and the function can be elicited asking $O(tm)$ value queries according to the procedure described in the proof of Theorem 8 of [22]. □

Theorem 10. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{RO_{+M}} \bigcup \mathbf{Tool_t}$, with t polynomial in m. Then, the elicitor can learn f correctly in polynomial time by asking at most $\frac{m(m+1)}{2} + O(tm)$ value queries.*

Proof. The elicitor asks the value of every singleton. Then, the elicitor asks the value of every two-item bundle, and it starts building the second level of $H_I(f)$. If $f \in \mathbf{RO_{+M}}$, then all the nodes in the second level of H_I must have weight ≤ 0. In particular, if the LCA of a and b in the read-once formula corresponding to f is a SUM gate, then we have $w(ab) = 0$; otherwise, the LCA is a MAX gate, and we have $w(ab) = -\min\{f(a), f(b)\}$. On the other hand, if $f \in \mathbf{Tool_t}$ all the weights in H_I must be positive. Thus, when the elicitor meets the first non-zero weight for a two-items bundle, it can determine whether $f \in \mathbf{RO_{+M}}$ or $f \in \mathbf{Tool_t}$, and can complete the elicitation of f by asking at most $\frac{m(m+1)}{2} + O(tm)$ value queries overall. If all the weights of the second level nodes in H_I are zero, the elicitor can safely conclude that $f \in \mathbf{Tool_t}$, and it continues elicitation by asking $O(mt)$ value queries overall. Note that the elicitor's decision is never wrong, since if $f \in \mathbf{RO_{+M}}$ and all the weights of the second level nodes of H_I are zero, then f must be the linear valuation function, which trivially belongs to $\mathbf{Tool_t}$ also. □

Theorem 11. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{G_2} \bigcup \mathbf{INT}$, and assume that, in case $f \in \mathbf{INT}$, we are in the scenario a) or b) of Theorem 2. Then, the elicitor can learn f correctly in polynomial time by asking at most $m^2 - m + 1$ value queries.*

Proof. Let us consider the more general case in which, in case $f \in \mathbf{INT}$, the elicitor does not know the linear order of the items, but it can build this order based on the valuation of the two-item bundles (scenario b) of Theorem 2). Note that $f \in \mathbf{INT}$ implies that all the weights associated to two-item bundles in H_I are positive. In fact, $f \in \mathbf{INT}$ implies $f(ab) \geq f(a) + f(b)$ for any ab. Thus, the elicitor can follow the following strategy. It first asks the valuation of every singleton; then, the valuation of every two-item bundles. During this process, the elicitor builds the G_2 graph. If G_2 contains at least one edge with negative weight, or it is not a line connecting the items, then the elicitor knows that $f \in \mathbf{G_2}$, and we are done. Otherwise, the elicitor knows that $f \in \mathbf{INT}$, and continues elicitation asking the value of the bundles which are compatible with the linear order induced by G_2, for a total of $m^2 - m + 1$ value queries (including the initial queries on singleton and two-item bundles). □

Theorem 12. *Assume that the elicitor only knows that the valuation function f to be learned belongs to* $\mathbf{Tool_t} \bigcup \mathbf{INT}$*, with t polynomial in m. Further, assume that, in case $f \in \mathbf{INT}$, we are in the scenario a) or b) of Theorem 2. Then, the elicitor can learn f correctly in polynomial time by asking at most $O(m(t+m))$ value queries.*

Proof. The elicitor starts asking the value of the singletons and two-item bundles. Depending on the outcomes to two-item bundles queries, the elicitor tries to build a linear order for the items. If any of the outcomes to the two-item bundles queries is not coherent with the linear order built so far, the elicitor concludes that $f \in \mathbf{Tool_t}$, and preference elicitation can be completed by asking $O(mt + m^2) = O(m(m+t))$ value queries overall. Otherwise, the elicitor is still not able to determine to which class f belongs. However, only two scenarios are possible:

- if $f \in \mathbf{INT}$, all the weights at levels greater than 2 in $H_I(f)$ are zero, except for those associated to bundles which are compatible with the linear order. The weights of these bundles might be negative.
- if $f \in \mathbf{Tool_t}$, at most t of the weights at levels greater than 2 in $H_I(f)$ can be non-zero. These weights can be in arbitrary positions, but are strictly positive.

The elicitor does not know which of the two scenarios actually holds, and it has to figure it out a way to conclude, asking value queries only (and only a polynomial number of them), which one of the two hypotheses actually holds. Note that if $f \in \mathbf{Tool_t} \bigcap \mathbf{INT}$ both of the conclusions the elicitor might take are correct.
In order to discover to which class f actually belongs, the elicitor asks the value of the bundles compatible with the linear ordering which has been previously computed. Based on the outcomes of these queries, the elicitor computes the weights of the corresponding nodes in $H_I(f)$ *under the assumption that* $f \in \mathbf{INT}$ (i.e., assuming that the remaining weights in the levels greater than 2 are 0). Then, it takes its decision according to the following rule: if there exists at least one strictly negative weight amongst the computed weights, then $f \in \mathbf{INT}$ and preference elicitation is done (asking $O(m^2)$ value queries overall); otherwise, the elicitor concludes that $f \in \mathbf{Tool_t}$, and it continues preference elicitation as described in the proof of Theorem 8 of [22] (note that all the minterms of cardinality at most 2 have already been discovered). In this case, the overall number of value queries is $O(m(m+t))$.
In the following, we prove that the elicitor never takes the wrong decision. First, assume that $f \in \mathbf{INT}$. Then, the weights calculated by the elicitor when it asks the value of the bundles compatible with the linear order are correct, i.e. they equal the actual weights of the valuation function f. If at least one of them is strictly negative, then f is correctly identified as belonging to the **INT** class. Otherwise, it must be $f \in \mathbf{Tool_t} \bigcap \mathbf{INT}$, and the elicitor's conclusion $f \in \mathbf{Tool_t}$ is again correct. In this case, the elicitor will ask further useless value queries

(those needed to identify the minterms of f which, given that $f \in \mathbf{Tool_t} \bigcap \mathbf{INT}$, can only be bundles compatible with the linear order). However, these useless queries are polynomially many, and polynomial time value query elicitation is not impaired.

Assume now that $f \in \mathbf{Tool_t} - \mathbf{INT}$ (we recall that the case $f \in \mathbf{Tool_t} \bigcap \mathbf{INT}$ is not a problem for the elicitor). Let us denote with $\mathcal{C}$ the set of bundles of cardinality at least 3 compatible with the linear order, and with $\mathcal{M}$ the set of minterms of f of cardinality at least 3. Note that, since $f \in \mathbf{Tool_t} - \mathbf{INT}$, we have $\mathcal{M} - \mathcal{C} \neq \emptyset$. For any $S \in \mathcal{C}$, let $w(S)$ denote the weight associated to node S in $H_I(f)$ as calculated by the elicitor. Since the elicitor calculates $w(S)$ under the wrong assumption that $f \in \mathbf{INT}$, this weight is in general different from the actual weight $w'(S)$ associated to S. In what follows we prove that, for any $S \in \mathcal{C}$ we have $w(S) \geq 0$. Since S is arbitrary, from this fact it immediately follows that the elicitor, after asking all the value queries associated to compatible bundles, correctly concludes that $f \in \mathbf{Tool_t}$, and we are done.

Let $h : \mathcal{M} \mapsto \mathcal{C}$ be the function that maps any bundle $\bar{S} \in \mathcal{M}$ into the bundle $S = h(\bar{S}) \in \mathcal{C}$ such that $\bar{S} \subseteq S$ and S has minimum cardinality amongst all the supersets of $\bar{S}$ in $\mathcal{C}$. It is easy to see that, for any $\bar{S} \in \mathcal{M}$, such a bundle S exists and is unique. In fact, if $\bar{S} \in \mathcal{M} \bigcap \mathcal{C}$, then $h(\bar{S}) = \bar{S}$. Otherwise, we must have $|\bar{S}| < m$, since the grand bundle is compatible with any linear order. This implies that for any $\bar{S} \in \mathcal{M}$, at least a superset $S \supseteq \bar{S}$ in $\mathcal{C}$ always exists. Let us now assume by contradiction that there exist distinct bundles $S_1, S_2 \in \mathcal{C}$ such that $|S_1| = |S_2| = c < m$, $\bar{S} \subset S_1$, $\bar{S} \subset S_2$, and $\bar{S} \not\subseteq S$, for any other $S \in \mathcal{C}$ of cardinality at most $c-1$. Since $\bar{S} \subset S_1$ and $\bar{S} \subset S_2$, we must have that $\bar{S} \subseteq S_1 \bigcap S_2 = S'$. On the other hand, $S_1, S_2 \in \mathcal{C}$ implies that $S' \in \mathcal{C}$ and, given that $|S_1| = |S_2| = c$ and $S_1 \neq S_2$, we have $|S'| < c$. This is a contradiction, since S' would be a compatible bundle of cardinality strictly smaller than c that contains $\bar{S}$.

We now prove that, for any $S \in \mathcal{C}$, we have:

$$w(S) - w'(S) = \sum_{\bar{S} \in h^{-1}(S) - S} w'(\bar{S}) \ , \tag{1}$$

where $h^{-1}(S)$ denotes the reverse image of S in $\mathcal{M}$, i.e., the set of all $\bar{S} \in \mathcal{M}$ such that $h(\bar{S}) = S$. From this, the theorem follows immediately by observing that $f \in \mathbf{Tool_t}$ implies that $w'(S') \geq 0$ for *any* bundle S'. The proof is by induction on $|S|$. If $|S| = 3$, then we have that $h^{-1}(S) = \emptyset$, thus the right hand term of (1) is 0. On the other hand, $|S| = 3$ implies that $w(S) = w'(S)$ (i.e., the weight calculated by the elicitor is always correct when $|S| = 3$); thus, the equality is satisfied. Let us now assume that the property holds for bundles of cardinality at most i. We have:

$$w(S) = f(S) - \sum_{S' \subset S, |S'| \leq 2} w'(S') - \sum_{S' \in \mathcal{C} | S' \subset S} w(S') \ ,$$

On the other hand, we have:

$$w'(S) = f(S) - \sum_{S' \subset S, |S'| \leq 2} w'(S') - \sum_{S' \in \mathcal{C} | S' \subset S} w'(S') - \sum_{\bar{S} \in \mathcal{M} - \mathcal{C} | \bar{S} \subset S} w'(\bar{S}) \ .$$

Combining the two equalities, we can write:

$$w(S) - w'(S) = - \sum_{S' \in \mathcal{C} | S' \subset S} (w(S') - w'(S')) + \sum_{\bar{S} \in \mathcal{M} - \mathcal{C} | \bar{S} \subset S} w'(\bar{S}) \ .$$

Applying the inductive hypothesis, we obtain:

$$w(S) - w'(S) = - \sum_{S' \in \mathcal{C} | S' \subset S} \left(\sum_{\bar{S} \in h^{-1}(S') - S'} w'(\bar{S}) \right) + \sum_{\bar{S} \in \mathcal{M} - \mathcal{C} | \bar{S} \subset S} w'(\bar{S}) \ .$$

Observing that each of the $\bar{S} \subseteq S'$ for some $S' \subset S$ appears in exactly one of the $h^{-1}(\cdot)$ sets, we can rewrite the preceding equality as:

$$w(S) - w'(S) = - \sum_{\bar{S} \in \mathcal{M} - \mathcal{C} | \bar{S} \in h^{-1}(S') - S', \text{ for some } S' \in \mathcal{C}, S' \subset S} w'(\bar{S}) + \sum_{\bar{S} \in \mathcal{M} - \mathcal{C} | \bar{S} \subset S} w'(\bar{S}) \ . \tag{2}$$

Let us now consider an arbitrary set $\bar{S} \in \mathcal{M} - \mathcal{C}$ such that $\bar{S} \subset S$. If $\bar{S} \in h^{-1}(S')$, for some $S' \in \mathcal{C}, S' \subset S$, then the corresponding weight appears both in the first and second summation with opposite sign, and it is canceled. Thus, the only remaining terms in the right hand side of (2) are those corresponding to minterms $\bar{S}$ such $\bar{S} \in h^{-1}(S) - S$, and we are done. □

Theorem 13. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{RO}_{+\mathbf{M}} \bigcup \mathbf{INT}$, and assume that, in case $f \in \mathbf{INT}$, we are in the scenario a) or b) of Theorem 2. Then, the elicitor can learn f correctly in polynomial time by asking $O(m^2)$ value queries.*

Proof. The elicitor first asks the value of every singleton. Then, it starts asking the value of the two-item bundles, in any order. We recall than, in case $f \in \mathbf{RO}_{+\mathbf{M}}$, $f(ab) = f(a) + f(b)$ if the LCA of a and b in the read-once formula associated to f is a SUM gate, while $f(ab) = \max\{f(a), f(b)\}$ if the LCA is a MAX gate. The corresponding weights on $H_I(f)$ are 0 and $-\min\{f(a), f(b)\}$, respectively. On the other hand, if $f \in \mathbf{INT}$, we have that $f(ab) \geq f(a) + f(b)$ if a and b are adjacent in the linear order, and $f(ab) = f(a) + f(b)$ otherwise. Thus, the weights associated to two-item bundles in $H_I(f)$ cannot be negative if $f \in \mathbf{INT}$.
Based on the above observation, the elicitor follows the following strategy. As long as it asks the value of the two-item bundles, it computes the corresponding weights on $H_I(f)$. The first time the elicitor encounters a non-zero weight, say $w(ab)$, it can safely take the right decision: $f \in \mathbf{RO}_{+\mathbf{M}}$ if $w(ab) < 0$, $f \in \mathbf{INT}$ if $w(ab) > 0$. In case $f \in \mathbf{INT}$, the elicitor continues preference elicitation according to its decision, asking $O(m^2)$ total value queries. If all the weights

associated to the two-item bundles in $H_I(f)$ are zero, the elicitor decides that $f \in \mathbf{INT}$, and continues elicitation accordingly. Note that if $f \in \mathbf{RO_{+M}}$ and all the weights of two-item bundles in H_I are 0, then f corresponds to the linear valuation function, which trivially belongs to **INT** also (since the linear valuation has no super-addivities between items). So, the elicitor never takes the wrong decision. □

Theorem 14. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{Tool_{-t}} \bigcup \mathbf{INT}$, where t is polynomial in m, and assume that, in case $f \in \mathbf{INT}$, we are in the scenario a) or b) of Theorem 2. Then, the elicitor can learn f correctly in polynomial time by asking $O(m(m+t))$ value queries.*

Proof. The elicitor asks the value of every singleton, and builds the first level of $H_I(f)$ accordingly. Then, the elicitor can resolve the uncertainty by asking the value of the grand bundle: if $f(I) < \sum_{a \in I} f(a)$, then it must be $f \in \mathbf{Tool_{-t}}$ (we recall that $f \in \mathbf{INT}$ implies that f is substitutability-free); otherwise, the elicitor can safely conclude that $f \in \mathbf{INT}$. Note that $f \in \mathbf{Tool_{-t}}$ and $f(I) = \sum_{a \in I} f(a)$ implies that f is the linear valuation function, which trivially belongs to **INT**. Once uncertainty is resolved, elicitation is continued according to whether $f \in \mathbf{INT}$ or $f \in \mathbf{Tool_{-t}}$, asking at most $O(m(m+t))$ queries overall. □

Theorem 15. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{Tool_{-t}} \bigcup \mathbf{G_2}$, where t is polynomial in m. Then, the elicitor can learn f correctly in polynomial time by asking at most $\frac{m(m+1)}{2} + O(tm)$ value queries.*

Proof. The proof is a straightforward modification of the proof of Theorem 9. □

Theorem 16. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{Tool_{-t_1}} \bigcup \mathbf{Tool_{t_2}}$, where t_1 and t_2 are polynomial in m. Then, the elicitor can learn f correctly in polynomial time by asking at most $m + O(t_{max}m)$ value queries, where $t_{max} = \max\{t_1, t_2\}$.*

Proof. The elicitor first ask the value of every singleton, then the value of the grand bundle. If $f(I) < \sum_{a \in I} f(a)$, then it must be $f \in \mathbf{Tool_{-t_1}}$; on the other hand, if $f(I) > \sum_{a \in I} f(a)$, then it must be $f \in \mathbf{Tool_{t_2}}$. In both cases, after the uncertainty is resolved the elicitor continues elicitation accordingly. Finally, we observe that $f(I) = \sum_{a \in I} f(a)$ and $f \in \mathbf{Tool_{-t_1}} \bigcup \mathbf{Tool_{t_2}}$ imply that f is the linear valuation function. □

Theorem 17. *Assume that the elicitor only knows that the valuation function f to be learned belongs to $\mathbf{Tool_{-t}} \bigcup \mathbf{RO_{+M}}$, where t is polynomial in m. Then, the elicitor can learn f correctly in polynomial time by asking at most $\frac{m(m+1)}{2} + O(tm)$ value queries.*

Proof. The elicitor asks the value of every singleton and two-item bundles, and compute the weights in the first two levels of $H_I(f)$ accordingly. If any of the weights in the second level of $H_I(f)$ is not compatible with the hypothesis $f \in \mathbf{RO_{+M}}$ (i.e., $w(ab) = 0$ or $w(ab) = -\min\{f(a), f(b)\}$), then the elicitor concludes $f \in \mathbf{Tool_{-t}}$, and completes elicitation asking $\frac{m(m+1)}{2} + O(tm)$ value queries overall. Otherwise, the elicitor builds the (unique) read-once formula F compatible with the weights in the first two-levels of $H_I(f)$ (here, we are assuming that the individual items have nonzero valuation). Observe that if F is such that all the sub-trees rooted at MAX gates in F have two leaves, then the elicitor can safely conclude that $f \in \mathbf{Tool_{-t}}$, and continues elicitation accordingly (in fact, all the weights of non singleton bundles in the hypercube representation of f are at most 0). Otherwise, asking the value of any bundle abc such that items a, b and c are leaves of a subtree rooted at a MAX gate is sufficient to solve the uncertainty: if $w(abc) > 0$, then it must be that $f \in \mathbf{RO_{+M}}$; otherwise, it must be that $f \in \mathbf{Tool_{-t}}$. This is because, if $f \in \mathbf{RO_{+M}}$, the weight associated to bundle abc in the hypercube representation of f equals $\min\{f(a), f(b), f(c)\} > 0$. □

We are now ready to present our generalized poly-time elicitor, GenPolyLearn. The algorithm, which is reported in Figure 5, is very simple: initially, the hypothesis set Hp contains all the five classes. After asking the value of every singleton, GenPolyLearn asks the value of every 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 18. *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.*

Proof. It is easy to see that all the decisions taken by the algorithm in steps 1–13 are correct. If after step 13 the size of Hp is still above 2, only the three cases listed in steps 14–37 are possible. Let us consider each case separately.

- *case* 1. In this case, we have Hp= $\{\mathbf{Tool_t}, \mathbf{G_2}, \mathbf{INT}\}$. If $f \in \mathbf{G_2}$, then all the weights at levels greater than 2 in $H_I(f)$ are zero. So, if it turns out that $f(I) \neq \sum_{S \subseteq I, |S| \leq 2} w(S)$, the algorithm can safely discard hypothesis $\mathbf{G_2}$ (step 20), and continues elicitation as in the proof of Theorem 12. On the other hand, if it turns out that $f(I) = \sum_{S \subseteq I, |S| \leq 2} w(S)$, then $\mathbf{Tool_t}$ can be safely excluded from the hypotheses set: in fact, $f \in \mathbf{Tool_t}$ and $f(I) = \sum_{S \subseteq I, |S| \leq 2} w(S)$ implies that $f \in \mathbf{G_2}$. So, elicitation is continued as in the proof of Theorem 11 (step 18).

Algorithm GenPolyLearn:

0. Hp=$\{\mathbf{RO_{+M}}, \mathbf{G_2}, \mathbf{Tool_t}, \mathbf{Tool_{-t}}, \mathbf{INT}\}$
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 $\mathbf{INT}$ 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 $\mathbf{INT}$ from Hp
12. until $|\text{Hp}| \leq 2$ or all the $w(ab)$ have been considered
13. if $|\text{Hp}| \leq 2$ then continue elicitation as described in theorems 8–17

otherwise:
14. *case 1*: all the $w(ab)$ weights are ≥ 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. 11
19. else
20. remove $\mathbf{G_2}$ from Hp
21. continue elicitation as in the proof of Th. 12
22. *case 2*: all the $w(ab)$ weights are ≤ 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. 17
27. else
28. remove $\mathbf{Tool_{-t}}$ from Hp
29. continue elicitation as in the proof of Th. 8
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 $\mathbf{INT}$, $\mathbf{Tool_t}$, $\mathbf{G_2}$, $\mathbf{RO_{+M}}$ 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. 12

Figure 5: 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.

– *case* 2. In this case, we have Hp= $\{\mathbf{Tool_{-t}}, \mathbf{G_2}, \mathbf{RO_{+M}}\}$. Again, if $f(I) \neq \sum_{S \subseteq I, |S| \leq 2} w(S)$, then $\mathbf{G_2}$ can be safely discarded, and elicitation is continued as in the proof of Theorem 17 (step 26). On the other hand, if it turns out that $f(I) = \sum_{S \subseteq I, |S| \leq 2} w(S)$, then $\mathbf{Tool_{-t}}$ can be safely excluded from the hypotheses set: in fact, $f \in \mathbf{Tool_{-t}}$ and $f(I) = \sum_{S \subseteq I, |S| \leq 2} w(S)$ implies that $f \in \mathbf{G_2}$. So, elicitation is continued as in the proof of Theorem 8 (step 29).

– *case* 3. In this case, we have Hp= $\{\mathbf{Tool_{-t}}, \mathbf{G_2}, \mathbf{RO_{+M}}, \mathbf{INT}, \mathbf{Tool_t}\}$. Again, asking the value of the grand bundle is sufficient to solve uncertainty: if $f(I) < \sum_{a \in I} f(a)$, then **INT** and $\mathbf{Tool_t}$ can be excluded from Hp, since these classes include only substitutability-free valuations. On the other hand, if $f \in \mathbf{G_2}$ and all the $w(ab)$ weights are zero, then f is the linear valuation function. This is incompatible with the fact that $f(I) < \sum_{a \in I} f(a)$, so also $\mathbf{G_2}$ can be removed from Hp. The same argument applies for $\mathbf{RO_{+M}}$: since all the $w(ab)$ weights are zero, all the inputs of the read-once formula have a SUM gate as LCA, i.e., f is the linear valuation function. So, at step 34, the algorithm correctly concludes that $f \in \mathbf{Tool_{-t}}$, and continues elicitation accordingly. On the other hand, if $f(I) \geq \sum_{a \in I} f(a)$, then the $\mathbf{Tool_{-t}}$ class can be safely discarded. In fact, given that $w(ab) = 0$ for every ab, the only possibility for a function f in $\mathbf{Tool_{-t}}$ to verify $f(I) \geq \sum_{a \in I} f(a)$ is to be the linear valuation function, which trivially belongs to any of the other classes. A similar argument applies to the $\mathbf{G_2}$ and $\mathbf{RO_{+M}}$ classes. Thus, the algorithm can safely remove $\mathbf{Tool_{-t}}$, $\mathbf{G_2}$ and $\mathbf{RO_{+M}}$ from Hp, and continues elicitation accordingly (step 37).

□

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.

We first observe that the results obtained in the related problem of learning concepts using membership queries (see, for instance, [1, 10, 12]) cannot be directly applied in our context. In fact, a concept in the machine learning theory is defined essentially in terms of its characteristic function: we have

a set of instances (bundles, using our terminology), and a certain concept is defined by specifying which of the instances satisfy the concept. Translated into our framework, this corresponds to allowing only boolean valuation functions, in which the value of a bundle S is 1 if S satisfies the target concept, and 0 otherwise. So, only *negative* results obtained in the concept learning theory can be useful in our framework (and we will actually use one of these results below). However, some of the notions and techniques used in concept learning, once properly adapted to the preference elicitation setting, turned out to be useful to our positive purposes.

Let us consider a certain class $\mathbf{C}$ of valuations. Essentially, we have seen that if $\mathbf{C}$ is "sufficiently structured", then valuations in $\mathbf{C}$ are easy to elicit under the assumption that the fact the function to elicit is in $\mathbf{C}$ is known to the elicitor. In other words, if the elicitor knows that a valuation has certain features, it can take advantage of this knowledge to infer the value of many other bundles without asking all the queries. If $\mathbf{C}$ is structured enough, polynomially many queries are sufficient to learn all the (exponentially many) values. In the following, we make this argumentation more formal.

Let $\mathbf{C}$ be a class of valuations, f any valuation in $\mathbf{C}$, and $A_{\mathbf{C}}$ an elicitation algorithm for $\mathbf{C}$[8]. 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* $V_{\mathbf{C}}(\mathcal{Q}(f))$. 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 8 (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:*

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

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

Definition 9 (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 $p(m)$.*

[8] 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.

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

Definition 10 (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 11 (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 12 (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 13 (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.* $\mathbf{C}$.

Clearly, we have $SNI_{\mathbf{C}} \subseteq NI_{\mathbf{C}}$. The following theorem shows that for some class of valuations $\mathbf{C}$ 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 19. *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 $\mathcal{Q}' = \{q(S_1), \ldots, q(S_k), q(\bar{S})\}$, 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 20. *Let* **C** *be an arbitrary class of valuations. If the size of* $SNI_{\mathbf{C}}$ *is super-polynomial in* m*, then* $\mathbf{C} \notin PQE$.

Proof. Assume by contradiction that there exists an elicitation algorithm that, given any function $f \in \mathbf{C}$ and a description for **C**, learns f asking polynomially many queries. This means that, for any such function f, there exists a set of queries $\mathcal{Q}$ of polynomial size such that, for all $S \in 2^I$, $S \in V_{\mathbf{C}}(\mathcal{Q}(f))$. This is true in particular for the strongly non-inferable bundles. On the other hand, denoting with $\bar{S}$ an arbitrary strongly non-inferable bundle and observing that $\mathcal{Q}$ has polynomial size by assumption, we have that $\bar{S} \in V_{\mathbf{C}}(\mathcal{Q}(f))$ implies $q(\bar{S}) \in \mathcal{Q}$. Since there are super-polynomially many strongly non-inferable bundles, this would imply that the size of $\mathcal{Q}$ is not polynomial in m – contradiction. □

Theorem 20 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 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 21. *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* $\mathbf{RDNF} \notin PQE$.

Proof. The proof follows immediately by the fact that the **RDNF** class, whose strongly non-inferable set has size 0, is hard to elicit with value queries (see [1]). □

Theorem 21 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 **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 **C** poly-query elicitable?"

Furthermore, Theorem 21 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 19. In a certain sense, this class seems easier to elicit than a class **C** with $|SNI_{\mathbf{C}}|$ superpolynomial in m. In case of the class **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 $f \in \mathbf{RDNF}$ (it is the set $\mathcal{Q}'$ as defined in the proof of Theorem 19); 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 14 (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 15 (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 22. *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* $p(m)$*.*

Proof. Let us assume that $\mathbf{C}$ is in NPQE. Then, there exists a non-deterministic algorithm A which, given in input a description of $\mathbf{C}$ and chosen any function $f \in \mathbf{C}$, elicits f by asking at most polynomially many queries in at least one of the non-deterministic computations. Let $\mathcal{Q}$ be the set of queries asked by A in one of these computations. It is immediate to see that $\mathcal{Q}$ is a teaching set for f. Since such a polynomial size teaching set exists for any $f \in \mathbf{C}$, it follows that $TD(\mathbf{C})$ is polynomial in m.
Let us now assume that $TD(\mathbf{C})$ is polynomial in m. Then, for any $f \in \mathbf{C}$ there exists at least one teaching set of polynomial size. This teaching set corresponds to one of the non-deterministic computations of an algorithm A that asks the queries according to an arbitrary order; this computation, which asks polynomially many queries, ends after all the queries in the teaching set have been asked (here, we assume that A is a "smart" algorithm). So, for any $f \in \mathbf{C}$ at least one of the non-deterministic computations of A ends after asking polynomially many queries, i.e., $\mathbf{C} \in$ NPQE. □

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

Proposition 2. $PQE \subset NPQE$*.*

6 Conclusions and future research

In this paper we have investigated in detail the problem of preference elicitation with value queries in CAs. It is known that efficient elicitation in this setting is not feasible unless some restrictions one the bidders' valuations are imposed. We have considered several classes of structured valuations in which efficient elicitation is possible, and proved that if $\mathbf{C_1}$ and $\mathbf{C_2}$ are classes of "easy to elicit with value queries" valuations, also the class $\mathbf{C_1} \bigcup \mathbf{C_2}$ has this property. This result concerns communication complexity, and it does not consider the computational complexity of identifying the queries to ask. We have proved that there exist classes of "easy to elicit with value queries" valuations $\mathbf{C_1}, \mathbf{C_2}$ such that eliciting a function in $\mathbf{C_1} \bigcup \mathbf{C_2}$ can be done asking polynomially many queries, but identifying the queries to ask is NP-complete. Despite this hardness result, we have presented an efficient polynomial time elicitation algorithm which, given any valuation function f in $\mathbf{RO_{+M}} \bigcup \mathbf{Tool_{-t}} \bigcup \mathbf{Tool_t} \bigcup \mathbf{G_2} \bigcup \mathbf{INT}$, learns f correctly.

We have also made some considerations on what renders a certain class of valuations "easy to elicit with value queries", proving a necessary condition for efficient elicitation with value queries. We have also introduced the concept of non-deterministic poly-query elicitation, and proved that a class of valuations is non-deterministically poly-query elicitable if and only if its teaching dimension is polynomial in the number of items on sale.

Overall, we believe the results presented in this paper constitute a significant step forward towards the characterization of efficient preference elicitation with value queries in CAs. Nevertheless, several issues remain open, stimulating further research on this topic. In particular, the problem of identifying sufficient conditions for making a class of valuations "easy to elicit with value queries" remains open.

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", *Journal of Machine Learning Research*, Vol. 5, pp. 649–667, 2004. A shorter version of this paper appeared in Proc. 16th 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] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan, D. Wilkins, "How Many Queries Are Needed to Learn?", *Journal of the ACM*, Vol. 43, n. 5, pp. 840–862, 1996.

[13] 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.

[14] S. Lahaie, D. Parkes, "Applying Learning Algorithms to Preference Elicitation", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 180–188, 2004.

[15] 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.

[16] J. MacKie-Mason, H.R. Varian, "Generalized Vickrey Auctions", *working paper*, Univ. of Michigan, 1994.

[17] N. Nisan, I. Segal, "The Communication Requirements of Efficient Allocations and Supporting Lindhal Prices", *internet draft*, version March 2003.

[18] M.H. Rothkopf, A. Pekec, R.H. Harstad, "Computationally Managable Combinatorial Auctions", *Management Science*, Vol. 44, n. 8, pp. 1131–1147, 1998.

[19] T. Sandholm, S. Suri, "BOB: Improved Winner Determination in Combinatorial Auctions and Generalizations", *Artificial Intelligence*, Vol. 145, pp. 33–58, 2003.

[20] T. Sandholm, "Algorithm for Optimal Winner Determination in Combinatorial Auctions", *Artificial Intelligence*, Vol. 135, pp. 1–54, 2002.

[21] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders", *Journal of Finance*, Vol. 16, pp. 8–37, 1961.

[22] 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.

[23] E. Zurel, N. Nisan, "An Efficient Approximate Allocation Algorithm for Combinatorial Auctions", *Proc. 3rd ACM Conference on Electronic Commerce (EC)*, pp. 125–136, 2001.