Combinatorial Auctions with k-wise Dependent Valuations (Extended Abstract)*

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

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

Abstract. Combinatorial auctions have attracted the attention of many researchers in recent years, due to their potential to significantly increase the economic efficiency of auctions. Unfortunately, the implementation of this type of auctions poses several challenges, including efficient computation of the optimal allocation of goods, and efficient communication of the bidders' preferences to the auctioneer. It is known that both these problems are hard to solve, unless some restrictions on the space of the possible bidders' preferences (also called valuations) are imposed.
In this paper, we analyze the computational and communication complexity of combinatorial auctions from a new perspective: the degree of mutual interdependency between the items on sale in the bidders' preferences. More specifically, denoting with $\mathbf{G_k}$ the class of valuations displaying up to k-wise dependencies, we consider the hierarchy $\mathbf{G_1} \subset \mathbf{G_2} \subset \cdots \subset \mathbf{G_m}$, where m is the number of items on sale. We show that the minimum non-trivial degree of interdependency (2-wise dependency) is sufficient to render hard the problem of computing the optimal allocation. On the other hand, bidders' preferences can be communicated efficiently (i.e., exchanging a polynomial amount of information) as long as the interdependencies between items are limited to sets of cardinality up to k, where k is an arbitrary constant. The amount of communication required to transmit the bidders' preferences becomes super-polynomial (under the assumption that only value queries are allowed) when interdependencies occur between sets of cardinality $g(m)$, where $g(m)$ is an arbitrary function such that $g(m) \to \infty$ as $m \to \infty$.
Besides proving these results, in this paper we analyze in detail the class of 2-wise dependent valuations. We prove that there exist restricted cases of such valuations for which computing the optimal allocation is easy. Furthermore, we consider the problem of approximate elicitation, in which the auctioneer learns, asking polynomially many value queries,

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

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

an approximation of the bidders' actual preferences. Most of the results on 2-wise dependent valuations are generalized to the case of k-wise dependent preferences, where k is an arbitrary constant.

1 Introduction

Background. Combinatorial auctions (CAs) have recently emerged as a mechanism to improve economic efficiency when many items are on sale. CAs can be used, for instance, to sell spectrum licenses, pollution permits, land lots, and so on [5]. In a CA, bidders can present bids on bundles of items, and thus may easily express complementarities (i.e., the bidder values two items together more than the sum of the valuations of the single items), and substitutabilities (i.e., the two items together are worth less than the sum of the valuations of the single items) between the objects on sale[3]. The function that, given a bundle, returns the bidder's value for that bundle is called *valuation function*, or simply valuation.

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 the bidders' preferences (valuation functions) to the auctioneer.

Historically, the first problem that has been addressed in the literature is winner determination. In [11], it is shown that solving the winner determination problem is NP-hard, and in [13] it is proved that even computing a good approximation of the optimal allocation is NP-hard (unless NP=ZPP). 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 [3]. Several elicitation algorithms, based on different type of queries (e.g., rank, order, or value queries), have been proposed [3, 4, 6]. Unfortunately, a recent result by Nisan and Segal [9] shows that elicitation algorithms have no hope of considerably reducing the communication complexity in the worst case. In fact, 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.

Given these computational and communication difficulties, several authors have presented restricted CA settings, in which solving either the winner determination problem, or eliciting bidders' preferences, or both, are easy [1, 2, 7, 8, 10–15]. The challenge here is to identify classes of valuations which are both sufficiently general (in the sense that they allow to express super-, or sub-additivity, or both, between items) and realistic.

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

Motivation. In this paper, we analyze the complexity of winner determination and preference elicitation in CAs from a new perspective: the degree of mutual interdependency between the items on sale. In general, a set of items displays some form of interdependency when their value as a bundle is different from the sum of their values as single items, resulting in complementarity or substitutability between the objects. Although this can be thought of as the distinguishing feature of CAs, to the best of our knowledge no research has studied the effect of the degree of this interdependency on the computational or communication complexity.

The degree of mutual interdependency between objects is clearly related to the computational and communication efficiency of CAs. If the items marginal values are independent of each other (or, if independence holds between items), then the bidders' preferences are linear (i.e., the valuation of a bundle is the sum of the values of the items it contains); in this situation, computing the optimal allocation is straightforward, and communication complexity is not an issue. However, this case does not require a CA setting, since the items could be auctioned sequentially with the same economic efficiency. On the other hand, when the degree of mutual interdependency is maximal (i.e., up to m-wise dependencies are displayed in the bidders' valuations, where m is the number of items on sale), we have fully general valuations, and both winner determination and preference elicitation are hard. So far, to our best knowledge, nothing is known about the complexity of the winner determination and preference elicitation when the degree of interdependency is somewhere between 1 and m.

In this paper, we consider CA settings in which the bidders' preferences display up to k-wise dependency between items, where $1 < k < m$. We believe that this type of valuation is likely to arise in many economic scenarios. For instance, when the items on sale are related to a geometric or geographic property (e.g., spectrum frequencies, railroad tracks, land slots,...), it is reasonable to assume that only items that are geometrically/geographically close display some form of interdependency. Another consideration that motivates our interest in k-wise dependent valuations is that, due to cognitive limitations, it might be difficult for a bidder to understand the inter-relationships between large group of objects.

Summary of results. In this paper, we show that the minimum non-trivial degree of interdependency between items (2-wise dependency) is sufficient to render the winner determination problem hard to solve. On the other hand, bidders' preferences remain easy to elicit as long as k is an arbitrary constant. The preference elicitation problem becomes hard as soon as $g(m)$-wise dependencies between objects are allowed, where $g(m)$ is an arbitrary function such that $g(m) \to \infty$ as $m \to \infty$.

Besides proving the results stated above, in this paper we analyze in details the class of 2-wise dependent valuations. This class is rich enough to express both complementarities and substitutabilities between items, and even costly disposal of items. Under this respect, 2-wise dependent valuations are more expressive than the classes of easy to elicit valuations introduced so far [1, 9, 15], which are

based on the free disposal assumption. First, we consider the situation in which the preferences are "almost 2-wise dependent", and we prove that in this case the auctioneer can learn a valuation function that is "close" to the original one asking polynomially many queries. Even better, we prove that if the valuation function is strongly super-modular (see Section 4 for a formal definition of this property), then the approximation error that we have if we ask only $\frac{m(m+1)}{2}$ out of all the 2^m possible queries can be bounded in a non-trivial way. This result is interesting, because strongly super-modular valuations are hard to elicit. So, our result proves that by asking a negligible fraction of queries, we can learn a valuation function that is close to the original valuation. Then, we consider the complexity of allocation, proving that there exists a restricted class of 2-wise dependent valuations for which solving the winner determination problem is easy. This restricted class is still powerful enough to express costly disposal, and it is easy to allocate and elicit. To the best of our knowledge, this is the first non-trivial CA setting with these features. Most of the results holding for 2-wise dependent valuations are generalized to the case of k-wise dependent valuations, where k is an arbitrary constant.

Remark. Before ending this section we want to remark that in this paper we focus our attention on a restricted case of preference elicitation, in which the elicitor can ask only *value queries* (what is the value of a particular bundle?) to the bidders. Our interest in value queries is due to the fact that, from the bidders' point of view, these queries are very intuitive and easy to understand.

2 2-wise dependent valuations

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 $v : 2^I \mapsto \mathbb{R}^+$ that assigns to any bundle $S \subseteq I$ its valuation. To make the notation less cumbersome, in this paper we will use notation $a, b, \ldots$ to denote singletons, $ab, bc, \ldots$ to denote two-item bundles, and so on.

Let us consider a bidder (agent) participating in the auction, issuing bids on bundle of items. In the following, we will focus on the valuation function of an arbitrary bidder, denoted A.

In general, the agent A will be interested in a subset of the items on sale, denoted I_A. Denoting with $v(a)$ the valuation of the singleton item, for every $a \in I_A$, in general we might have $v(a) = 0$ for some of them. These items, called *zero value items*, do not have a value in themselves, but they can be used to form valuable bundles with other items.

Let us consider an arbitrary pair a, b of items in I_A. We classify every such pair as *independent* (or *additive*), *super-additive* or *sub-additive* depending on how the valuation of the bundle ab relates to $v(a) + v(b)$. In particular, we have:

- if $v(ab) = v(a) + v(b)$ then a and b are independent items;
- if $v(ab) > v(a) + v(b)$ then a and b are super-additive items;
- if $v(ab) < v(a) + v(b)$ then a and b are sub-additive items;

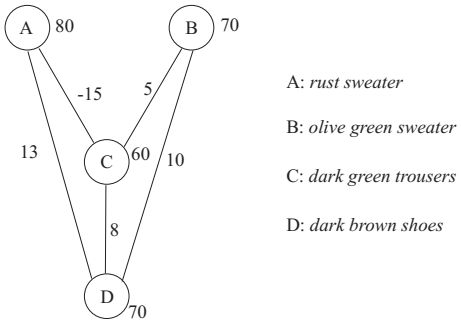

Fig. 1. 2-wise dependency graph representing the bidder's valuation in the auction of fashion clothing.

Given the dependencies between any pair of items in I_A, the *2-wise dependency graph* G_2 can be constructed as follows:

- let there be a node for every item in I_A;
- label node a with $v(a)$;[4]
- if a and b are super- or sub-additive, put an (undirected) edge (a, b) in the graph, and label the edge with $v(ab) - (v(a) + v(b))$.

Under the assumption that there exist only 2-wise item dependencies in the valuation function of agent A, the G_2 graph can be used to calculate the valuation of any possible subset S of I_A 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 a 2-wise dependent valuation could be the following. Consider an auction of fashion clothing. In this scenario, it seems reasonable to assume that items display super- of sub-additivity depending on how good they look together. In Figure 1, there are four items on sale: a rust sweater, an olive green sweater, dark green trousers, and a pair of dark brown shoes. The items have values as singletons (e.g., the rust sweater is worth \$80), and show 2-wise dependencies when bundled together. For instance, the bundle composed by the olive green sweater, dark green trousers and dark brown shoes has a super-additive valuation (\$223 instead of \$200), because these items together form a nice outfit. Conversely, the rust sweater and the dark green trousers clash, so their value as a bundle is sub-additive (\$125 instead of \$140). Note that the sum of the values of the rust sweater, dark green trousers, and dark brown shoes is higher than the sum of the values of the olive green sweater, dark green trousers, and dark brown shoes bundle (\$210 and \$200, respectively). However, when considered as a bundle, the valuation of the former combination is lower than that of the second one (\$213 and \$223, respectively), due to the fact that the items in this combination do not form a good outfit (the rust sweater and the dark green trousers clash)[5].

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

3 Expressive power of G_2 valuations

2-wise dependent valuations can be elicited by asking $\frac{m(m+1)}{2}$ value queries (m single item queries, and $\frac{m(m-1)}{2}$ queries for the two item bundles). On the other hand, not all possible preferences can be expressed using G_2 graphs. So, it is interesting to compare the expressive power of 2-wise dependent valuations to that of other classes of valuations, such as those presented in [15], which can also be elicited asking a polynomial number of value queries.

Due to space limitations, most of the material contained in this section has been moved to the Appendix.

Remark. *Costly disposal* can be easily expressed using 2-wise dependencies graphs. Costly disposal models those situations in which the bidder incurs a cost for disposing undesired items. Thus, the monotonicity assumption typical of the *free disposal* setting, i.e. that $v(S') \geq v(S)$ for any $S' \supseteq S$, no longer holds. For instance, the fact that the bidder values a at 2 and b at 5, wants at most one of the items, and incurs a cost of 1 for disposing of an extra item, can be represented using the G_2 graph which assigns weight 2 to node a, 5 to node b, and weight -3 to the edge (a, b). To the best of our knowledge, 2-wise dependent valuations are the only known class of valuation functions that express costly disposal and can be elicited asking a polynomial number of queries. In fact, the classes of easy to elicit valuations defined in [1, 9, 15], as well as the preference elicitation techniques proposed in [6] and referred therein, are based on the free disposal assumption.

4 Learning almost 2-wise dependent valuations

In the following, we will denote with $\mathbf{G_2}$ the class of valuation functions that can be expressed using a G_2 graph. In this section, we consider the case in which the valuation function does not belong to $\mathbf{G_2}$, but it can be well approximated by a 2-wise dependent valuation v'.

In the remainder of this paper, we will make extensive use of the following representation of valuation functions. Given the set I of items on sale, 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.

[5] The fact the two sweaters have additive valuation is for the purpose of illustration only. In general, the two sweaters might display sub-additive valuation as well. However, in our example we want to emphasize that some of the items on sale might have additive valuation when bundled together. The independence is reasonable if the agent has to buy two sweaters (from some source) anyway, because winter is coming.

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

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

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

In the following, we will use the concept of distance of a valuation function from a class, which is formally defined as follows.

Definition 1 *Let v be an arbitrary valuation function, and $\mathbf{C}$ be an arbitrary class of valuation functions. Given a function $v' \in \mathbf{C}$, we say that v' is a δ-approximation of v if $|v(S) - v'(S)| \leq \delta$ for every bundle S. The distance between v and $\mathbf{C}$, denoted $d(v, \mathbf{C})$, is defined as the* $\min\{\delta | \exists v' \in \mathbf{C}$ *such that v' is a δ-approximation of $v\}$.*

Assume that the valuation function v to be elicited is a δ-approximation of a 2-wise dependency function v'. The following theorem shows that a $O(m^2\delta)$-approximation of v can be learned asking $\frac{m(m+1)}{2}$ value queries.

Theorem 1 *Assume that the valuation function v is a δ-approximation of v', for some $v' \in \mathbf{G_2}$. Then, a function $g \in \mathbf{G_2}$ can be learned asking $\frac{m(m+1)}{2}$ value queries, such that for any bundle of items S,*

$$|v(S) - g(S)| \leq \delta \left(1 + \frac{|S|(|S|-1)}{2}\right) .$$

Proof. Due to space limitations, the proof of this theorem, as well as the proofs of all the theorems presented in this paper, is reported in the Appendix.

The bound stated in Theorem 1 is tight for the elicitation technique used in the proof, in the sense that, for any value of m, there exist valuation functions v, v', with $v' \in \mathbf{G_2}$ such that v' is a δ-approximation of v, and the function g learned in polynomial time is a $\delta\left(1 + \frac{m(m-1)}{2}\right)$-approximation of v. This is proved in the following theorem.

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

Theorem 2 *The bound on the approximation factor of the learned function g stated in Theorem 1 is tight.*

Let us consider valuations such that all the weights in the corresponding H_I graph are non-negative. We call these valuations *strongly super-modular* valuations[7]. It is not hard to see that strongly super-modular valuations are hard to elicit with value queries, because they require exponentially many values to specify. The following theorem gives an upper bound to distance between strongly super-modular valuations and the $\mathbf{G_2}$ class, which contains easy to elicit valuations.

Theorem 3 *Let v be an arbitrary strongly super-modular valuation, and let v_2 be the unique valuation function in $\mathbf{G_2}$ that coincides with v on the singletons and two-item bundles. Let $c_i(v) = \max_{S,|S|=i}\{v(S) - v_2(S)\}$, and let $M(v) = \max_{i=3,\ldots,m} \frac{2c_i(v)}{i(i+1)}$. Then, there exists a function $v' \in \mathbf{G_2}$ such that $|v(S) - v'(S)| \leq \frac{M(v)}{2} \cdot \frac{|S|(|S|+1)}{2}$ for any bundle S.*

Corollary 1 *Let v be an arbitrary strongly super-modular valuation. Then, we have $d(v, \mathbf{G_2}) \leq \frac{M(v)}{2} \cdot \frac{m(m+1)}{2}$.*

The following theorem shows that the bound stated in Theorem 3 is tight.

Theorem 4 *There exists a strongly super-modular valuation v such that $d(v, \mathbf{G_2}) = \frac{M(v)}{2} \cdot \frac{m(m+1)}{2}$.*

The results stated in theorems 1 and 3 can be combined into the following theorem, which gives an upper bound on the error that we have when an arbitrary strongly super-modular valuation v is approximated using a function in $\mathbf{G_2}$.

Theorem 5 *Let v be an arbitrary strongly super-modular valuation. Then, a function $g \in \mathbf{G_2}$ can be learned asking $\frac{m(m+1)}{2}$ value queries such that:*

$$|v(S) - g(S)| \leq \frac{M(v)}{2} \left(\frac{|S|(|S|+1)}{2}\right) \left(1 + \frac{|S|(|S|-1)}{2}\right)$$

for any bundle S, where $M(v)$ is defined as in the statement of Theorem 3.

Although the bound on the approximation error stated in Theorem 5 is considerable, it is interesting that if v has a certain property (which is not sufficient to make it easy to elicit), then the approximation error that we have if we ask only $\frac{m(m+1)}{2}$ out of the $2^m - 1$ possible value queries can be bounded in a non-trivial way.

[7] The reason for this name is the following. If a valuation has the property that all the weights in the correspondent H_I graph are non-negative, then it is super-modular. On the other hand, there exist super-modular valuations such that some of the weights in the corresponding hypercube are strictly negative. Super-modular valuations are valuations with increasing marginal utility.

The approximation bound of Theorem 5 is composed of two terms: the first term, $\frac{M(v)}{2} \cdot \frac{|S|(|S|+1)}{2}$, is due to the fact that v in general is this far away from $\mathbf{G_2}$ valuations; the second term, $\left(1 + \frac{|S|(|S|-1)}{2}\right)$, derives from the fact that the elicitor does not know which is the function $v' \in \mathbf{G_2}$ that best approximates v. Thus, the elicitor can only *guess* a function $g \in \mathbf{G_2}$, and guessing the function costs at most an $\left(1 + \frac{|S|(|S|-1)}{2}\right)$ error. While the first term in the approximation error in general cannot be improved, since it derives from the fact that $v \notin \mathbf{G_2}$, a natural question is whether the elicitor might do a better guess than function g (which, we recall, corresponds to function v_2 as defined in the statement of Theorem 3). We believe that answering this question is not trivial, and we leave this as an open problem.

5 Allocation with $\mathbf{G_2}$ valuations

In this section, we investigate the computational complexity of the winner determination problem when all the bidders participating in the auction have $\mathbf{G}_2$ valuation functions.

Theorem 6 *Computing the optimal allocation in a CA where all the bidders have 2-wise dependent valuation functions is NP-hard, even when each bidder places only values of 0 on individual items, and places nonzero values on only two (adjacent) edges (in fact, a value of* 1 *on each of these edges).*

However, if the graph obtained merging the G_2 graphs of the bidders displays some structure, then the auction can be cleared in polynomial time.

Theorem 7 *Consider the graph of all vertices (items), and all edges between items such that at least one bidder places a nonzero value on the edge. Suppose this graph has no cycles (it is a forest). Then the optimal allocation can be computed in $O(nm)$ time, where n is the number of bidders.*

Note that Theorem 7 defines a non-trivial class of costly disposal valuation functions which can be elicited using a polynomial number of value queries, and for which the winner determination problem can be solved in polynomial time. To the best of our knowledge, this is the first class of computational and communication efficient costly disposal valuations which has been defined in the literature.

6 Generalization: k-wise dependency

The 2-wise dependency model can be easily extended to the case of k-wise dependency, for some $k \leq m$, by adding to the graph j-multiedges between subsets of the items of cardinality j, for any $j = 3, \ldots, k$. These multiedges account for up to k-wise dependencies between items. Given the k-wise dependency graph G_k, the valuation of a bundle S is obtained by considering the subgraph G^S induced by nodes in S, and summing up the weights of the nodes and of the

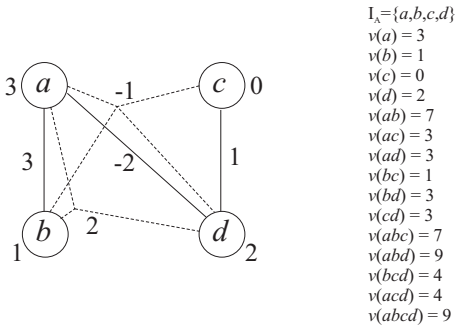

Fig. 2. Example of 4-wise dependence graph, and the corresponding valuation function v. Multiedges are represented with dashed lines.

edges (including all multiedges) in G^S. An example of G_4 graph, along with the corresponding valuation function, is shown in Figure 2. The class of valuations that can be expressed using a G_k graph is denoted $\mathbf{G_k}$.

The following theorem shows that if a valuation function is included in $\mathbf{G}_k$, for some constant $k < m$, then it can be elicited in polynomial time.

Theorem 8 *Let k be an arbitrary constant, and assume that there exist only up to k-dependencies between items in the valuation function v. Then, v can be elicited asking $O(m^k)$ value queries.*

The following theorems generalize some of the results presented in the previous sections to the case of k-wise dependent valuations.

Theorem 9 *Assume that the valuation function v is a δ-approximation of v', for some $v' \in \mathbf{G_k}$, with k an arbitrary positive constant. Then, a function $g \in \mathbf{G_k}$ can be learned asking $O(m^k)$ value queries, such that $v(S) = g(S)$ for any bundle S with $|S| \leq k$, and*

$$|v(S) - g(S)| < \delta \left(1 + \binom{|S|}{k}\right)$$

for any bundle S with $|S| > k$.

Theorem 10 *Let v be an arbitrary strongly super modular valuation, and let v_k be the unique valuation function in $\mathbf{G_k}$ that coincides with v on the bundles of cardinality at most k. Let $c_i(v) = \max_{S,|S|=i}\{v(S) - v_k(S)\}$, and let $M(v) = \max_{i=k+1,\ldots,m} \frac{c_i(v)}{\sum_{j=1\ldots k}\binom{i}{j}}$. Then, there exists a function $v' \in \mathbf{G_k}$ such that $|v(S) - v'(S)| \leq \frac{M(v)}{2} \cdot \sum_{j=1\ldots k}\binom{|S|}{j}$ for any bundle S.*

Corollary 2 *Let v be an arbitrary strongly super modular valuation. Then, we have $d(v, \mathbf{G_k}) \leq \frac{M(v)}{2} \cdot \sum_{j=1\ldots k}\binom{m}{j}$.*

Apparently, the bound stated in Theorem 10 seems to be looser than the one reported in Theorem 3, which would be counterintuitive. However, we have to consider that, denoting with $M_2(v)$ and $M_k(v)$ the value of M as in the statement of theorems 3 and 10, respectively, it is very likely to be $M_2(v) \gg M_k(v)$ in practice. In any case, denoting with $d_2, d_3, \ldots, d_k$ the distance between v and the $\mathbf{G_2}, \mathbf{G_3}, \ldots, \mathbf{G_k}$ classes, respectively, we have $d_2 \geq d_3 \geq \ldots d_k$ because the classes subsume each other.

7 The $\mathbf{G_k}$ hierarchy

Let $\mathbf{G}_k$ denotes the class of k-wise dependent valuations. It is clear that these classes define a hierarchy, where $\mathbf{G}_i \subset \mathbf{G}_{i+1}$ and every inclusion is strict. The bottom class of the hierarchy is the $\mathbf{G_1}$ class, which corresponds to the class of linear valuations (i.e., the valuation of any bundle is simply the sum of the valuations of the singletons). These valuations are easy to elicit and to allocate, but are of no interest in the CA setting. Let us consider the second element of the hierarchy, $\mathbf{G_2}$. Theorem 6 shows that valuations in this class are hard to allocate. This means that even the most limited form of interdependency between items (2-wise dependency), is sufficient to render the problem of finding the optimal allocation hard. On the other hand, valuations that display this limited form of item dependency remain easy to elicit. Indeed, Theorem 8 shows that elicitation remains easy as long as the interdependencies between items are limited to sets of cardinality k, where k is an arbitrary constant. What happens when the interdependencies are between sets of $g(m)$ objects, where $g(m)$ is an arbitrary function such that $g(m) \to \infty$ as $m \to \infty$? The following theorem shows that in this scenario preference elicitation with value queries becomes hard.

Theorem 11 *Let v be an arbitrary valuation in $\mathbf{G_{g(m)}}$, where $g(m)$ is an arbitrary function such that $g(m) \to \infty$ as $m \to \infty$. Then v is hard to elicit with value queries.*

Finally, the theorem below shows that the class at the top of this hierarchy, $\mathbf{G}_m$, is fully expressive, i.e., it can express any valuation function.

Theorem 12 *Every valuation function can be represented using the G_m model. Moreover, the representation of any valuation function is unique.*

Thus, we can end this section with the following theorem:

Theorem 13 *We can define a hierarchy on valuations with respect to the maximal interdependencies between the items: valuations which display up to k-wise dependencies belong to the $\mathbf{G_k}$ class. We have:*

$$\mathbf{G_1} \subset \mathbf{G_2} \subset \cdots \subset \mathbf{G_m} \ ,$$

where every inclusion is strict. The bottom class of the hierarchy, $\mathbf{G_1}$, corresponds to linear valuations, which are easy to elicit and allocate. Valuations in

$\mathbf{G_k}$, *where* $k \geq 2$ *is an arbitrary constant, are easy to elicit and hard to allocate. Valuations in* $\mathbf{G_{g(m)}}$, *where* $g(m)$ *is an arbitrary function such that* $g(m) \to \infty$ *as* $m \to \infty$, *are hard to elicit with value queries and hard to allocate. The class at the top of the hierarchy,* $\mathbf{G_m}$, *contains all possible valuations.*

References

1. A. Blum, J. Jackson, T. Sandholm, M. Zinkevic, "Preference Elicitation and Query Learning", *in Proc. Conference on Computational Learning Theory (COLT)*, 2003.
2. Y. Chang, C.S. Li, J. Smith, "Searching Dynamically Bundled Goods with Pairwise Relations", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 135–143, 2003.
3. W. Conen, T. Sandholm, "Preference Elicitation in Combinatorial Auctions", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 256–259, 2001.
4. W. Conen, T. Sandholm, "Partial-Revelation VCG Mechanisms for Combinatorial Auctions", *Proc. National Conference on Artificial Intelligence (AAAI)*, pp. 367–372, 2002.
5. S. de Vries, R. Vohra, "Combinatorial Auctions: a Survey", *INFORMS J. of Computing*, to appear.
6. B. Hudson, T. Sandholm, "Using Value Queries in Combinatorial Auctions", *poster presentation at ACM Conference on Electronic Commerce (EC)*, 2003.
7. 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.
8. N. Nisan, "Bidding and Allocation in Combinatorial Auctions", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 1–12, 2000.
9. N. Nisan, I. Segal, "The Communication Requirements of Efficient Allocations and Supporting Lindhal Prices", internet draft, version March 2003.
10. D. Parkes, "i-Bundle: An Efficient Ascending Price Bundle Auction", *Proc. ACM Conference on Electronic Commerce (EC)*, pp. 148–157, 1999.
11. M.H. Rothkopf, A. Pekec, R.H. Harstad, "Computationally Managable Combinatorial Auctions", *Management Science*, Vol. 44, n. 8, pp. 1131–1147, 1998.
12. T. Sandholm, S. Suri, "BOB: Improved Winner Determination in Combinatorial Auctions and Generalizations", *Artificial Intelligence*, Vol. 145, pp. 33–58, 2003.
13. T. Sandholm, "Algorithm for Optimal Winner Determination in Combinatorial Auctions", *Artificial Intelligence*, Vol. 135, pp. 1–54, 2002.
14. P.R. Wurman, M.P. Wellman, "AkBA: A Progressive, Anonymous Price Combinatorial Auction", *Proc. 2rd ACM Conference on Electronic Commerce (EC)*, pp. 21–29, 2000.
15. 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.
16. E. Zurel, N. Nisan, "An Efficient Approximate Allocation Algorithm for Combinatorial Auctions", *Proc. 3rd ACM Conference on Electronic Commerce (EC)*, pp. 125–136, 2001.

A Appendix

A.1 Expressive power of G_2 valuations

In this section, we compare the expressive power of 2-wise dependent valuations with the class of "easy to elicity" valuations presented in [15].

In [15], two types of valuation functions are considered: read-once formulas and Toolbox DNF.

Let us first consider read-once formulas. 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. As an example, the read-once formula $\text{ATLEAST}_2(2a, MAX(3b, c), d)$ gives value 0 to a, value 3 to ad, and value 5 to abc.

In [15], it is observed that read-once formulas can express both complementarities and substitutabilities. However, only "extreme" cases of super- and sub-additivity can be expressed. In particular, the MAX_c operator can be used to express "c most valued" sub-additivity: the valuation of the bundle composed by c' of the inputs to the gate equals the sum of the c items with highest value in the bundle. If $c' > c$, the resulting valuation is sub-additive. For instance, the read-once formula $\text{MAX}_2(2a, 3b, c)$ gives value 2 to a, value 3 to b, and value 1 to c, but only value 5 to the bundle abc. A similar observation applies to the super-addivity that can be expressed by read-once formulas. The super-additive operator is ATLEAST_c. However, only "c out of c'" super-additivity can be expressed: the valuation of a bundle is non-zero only if the bundle is composed by at least c of the c' inputs of the gate. So, the items do not have value as singletons, but they have value only when they are bundled with at least other $c-1$ items. For instance, $\text{ATLEAST}_2(2a, 3b, c)$, gives value 0 to singleton bundles a, b and c, and gives the super-additive value of 5 to the bundle ab.

From this discussion, it is clear that many natural sub- and super-additive valuations cannot be expressed by read-once formulas, while they can be easily expressed in the 2-wise dependency graph model. Intuitively, the super- and sub-additivities that cannot be expressed by read-once formulas are "marginal value" additivities, in which the items have a non zero value as singletons, and when bundled together they generate a positive or negative marginal utility. For instance, consider the valuation function v such that $v(a) = 2$, $v(b) = 3$, and $v(ab) = 6$. Function v can be easily expressed in the G_2 model (node a with label 2, node b with label 3, and edge (a, b) with label 1), while it cannot be expressed using read-once formulas. Similarly, valuation functions with negative marginal utilities, such as $v'(a) = 2$, $v'(b) = 3$ and $v'(ab) = 4$, can be easily expressed in G_2 but not with read-once formulas.

On the other hand, there exist valuation functions that can be expressed with read-once formulas but not with the G_2 model. In particular, formulas containing MAX_c or ATLEAST_c operators with at least 3 inputs cannot be expressed using G_2 (this easily follows from the fact that three-wise dependencies cannot be expressed in G_2). Note that, even if all the gates in the read once formula have at most two inputs, this is not sufficient for the resulting valuation function to be representable using G_2. For instance, the valuation function generated by the formula $\text{ATLEAST}_2(\text{MAX}_1(3a, b), 5c)$ cannot be expressed using G_2. Indeed, the intersection between the class of read-once formula and 2-wise dependency graph can be characterized as follows:

Theorem 14 *A valuation function v generated by a read-once formula f can be expressed using a 2-wise dependency graph if and only if the subtrees rooted at any of the* MAX_c *and* ATLEAST_c *gates have at most two inputs (items) amongst their leaves.*[8]

The immediate proof of this Theorem is omitted.

Let us now consider the other type of valuation functions considered in [15], i.e., ToolboxDNF. This class corresponds to the valuation functions that can be expressed as monotone polynomials. For instance, polynomial $3a+4ab+2bc+cd$ gives value 3 to item a, 0 to item b, value 9 to the bundle abc, and so on. Contrary to read-once formulas, ToolboxDNF cannot express substitutabilities between items, since only positive terms in the polynomial can be used. However, ToolboxDNF allows the representation of "marginal utility" super-additivities. For instance, function v such that $v(a) = 2$, $v(b) = 3$, and $v(ab) = 6$ can be represented by the polynomial $2a + 3b + ab$.

The intersection between the class of G_2 and ToolboxDNF valuation functions is characterized in the following theorem, whose immediate proof is omitted:

Theorem 15 *A valuation function v generated by a ToolboxDNF polynomial p can be expressed using a 2-wise dependency graph if and only if all the monomials in p contain at most two variables (items).*

Suppose that all the monomials in p contain at most two variables, and that there are t monomials. Then, the valuation function corresponding to p can be elicited in $O(m^2)$ time using the G_2 graph; this is an improvement over the $O(mt) = O(m^3)$ bound proved in [15] (which, however, holds for general ToolboxDNF valuations).

[8] The *only if* here is intended as follows. If the subtrees rooted at any of the MAX_c and ATLEAST_c gates have at most two inputs (items) amongst their leaves, then it is possible to choose the weight on the leaves in such a way that the resulting valuation cannot be expressed using a 2-wise dependency graph.

A.2 Proofs

*Proof (***Proof of Theorem 1.***).* The learned function g is built as follows. First, we ask the value of any bundle of at most two items. Then, we built the G_2 graph using these values, and we calculate g accordingly. It is immediate that $v(S) = g(S)$ for any bundle S of at most two items.
Let us now consider a bundle S with more than two items. Given a set S with $|S| > 2$, the value of v on S can be written as follows: $v(S) = \sum_{S' \subset S, |S'| \leq 2} w(S') + \sum_{S' \subseteq S, |S'|>2} w(S') = v_{\leq 2}(S) + v_{>2}(S)$. On the other hand, we have $g(S) = g_{\leq 2}(S) + g_{>2}(S)$. Since $g \in \mathbf{G_2}$, $g_{>2}(S) = 0$; furthermore, we have $g_{\leq 2}(S) = v_{\leq 2}(S)$, from which we get $g(S) = v_{\leq 2}(S)$. Thus we can write $|v(S) - g(S)| = |v_{>2}(S)|$.

Since v is a δ-approximation of a 2-wise dependent valuation, we have $|v(S) - v'(S)| \leq \delta$, for some $v' \in \mathbf{G}_2$. Thus, we can write $|v_{\leq 2}(S) + v_{>2}(S) - v'_{\leq 2}(S)| \leq \delta$, which, assuming w.l.o.g. that $v(S) \geq v'(S)$, gives $v_{>2}(S) \leq \delta + (v'_{\leq 2}(S) - v_{\leq 2}(S))$. Thus, the problem is reduced to finding an upper bound to $|v'_{\leq 2}(S) - v_{\leq 2}(S)|$.
Let us consider an arbitrary bundle ab of two items. Since v is δ-approximated by v', we have $|v(ab) - v'(ab)| \leq \delta$. Since there are $\frac{|S|(|S|-1)}{2}$ distinct such pairs in S, and each of them contributes with at most δ error, we have $|v'_{\leq 2}(S) - v_{\leq 2}(S)| \leq \delta \frac{|S|(|S|-1)}{2}$, and the theorem follows.

*Proof (***Proof of Theorem 2.***).* Let us consider the valuation function v which gives value 0 to any singleton and two-items bundle, value $\delta \frac{i(i-1)}{2}$ to any bundle composed by i items, with $i = 3, \ldots, m-1$, and value $\delta \left(\frac{m(m-1)}{2} + 1 \right)$ to the grand bundle. The weights on $H_I(v)$ are 0 for the nodes at level 1 and 2, $\delta \frac{i(i-1)}{2}$ for nodes at level i with $i = 2k+1$, $1 \leq k \leq \lfloor \frac{m}{2} \rfloor - 1$, and $-\delta \frac{i(i-1)}{2}$ for nodes at level i with $i = 2k$, $2 \leq k \leq \lceil \frac{m}{2} \rceil - 1$. Finally, the weight of the grand bundle is $\delta \frac{m(m-1)}{2}$ (with the appropriate sign depending on whether m is odd or even) augmented by δ.
Let us consider the valuation function $v' \in \mathbf{G_2}$ such that the weights on the corresponding H_I graph are as follows: 0 on the first level nodes, δ on the second level nodes, and 0 on nodes at level i, with $i \geq 3$. It is easy to see that $|v(S) - v'(S)| \leq \delta$ for any bundle S. On the other hand, the learned function g coincides with v on the singletons and two-item bundles, i.e., it is the constant function that gives value 0 to any item. Thus, the approximation error of g is maximum on the grand bundle, and it corresponds to the bound $\delta \left(\frac{m(m-1)}{2} + 1 \right)$ stated in Theorem 1.

*Proof (***Proof of Theorem 3.***).* W.l.o.g., assume that $v(S) = 0$ for any S with $|S| \leq 2$. This way, $v_2 = 0$ on every bundle, and the presentation of the proof is simplified.

The intuition behind the proof is the following. A function in $\mathbf{G_2}$ has non-zero weights only on the levels 1 and 2 of H_I, while the original function v has possibly non-zero (positive) weights in every level greater than 2. The idea is to set the weights on the nodes in levels 1 and 2 in H_I in such a way that the error the we have on any possible bundle is minimum. This way, we will obtain a function $v' \in \mathbf{G_2}$ with the desired properties.
Let us consider an arbitrary bundle S_i composed by i items, and let $N^i_{\leq 2}$ denote the set composed by nodes in the levels 1 and 2 of $H_I^{S_i}$, the sub-hypercube of dimension i rooted at S_i. The cardinality of $N^i_{\leq 2}$ is $\frac{i(i+1)}{2}$, as it is the cardinality of the node sets corresponding to any other bundle composed by i items. If we would assign weight $\frac{2v(S_i)}{i(i+1)}$ to any node in $N^i_{\leq 2}$ in the function $v' \in \mathbf{G_2}$, we would get a zero error in the valuation of bundle S_i. Of course, different bundles induce different "desired weights" on the nodes in levels 1 and 2 of H_I. Let us consider an arbitrary node u in level 1 or 2 of H_I, and denote with min_u and max_u the minimum and maximum values of these desired weights, calculated over the weights induced by all possible bundles of cardinality at least 3 that contain u. If we set the weight of u to $\frac{max_u + min_u}{2}$, we have that the contribution to the approximation error *on any possible bundle* due to node u is at most $\frac{max_u - min_u}{2}$. The proof of the theorem then follows by observing that:

- since all the weights in the $H_I(v)$ are non-negative, we have $min_u \geq 0$ for any node u in level 1 or 2;
- defining $c_i(v)$ and $M(v)$ as in the statement of the theorem, we have $max_u \leq M(v)$ for any node u in level 1 or 2;
- for any bundle S, there are exactly $\frac{|S|(|S|+1)}{2}$ nodes in levels 1 and 2 of the sub-hypercube of dimension $|S|$ rooted at S, each contributing with error at most $\frac{M(v)}{2}$.

*Proof (***Proof of Theorem 4.).** Let v be defined as follows: $v(S) = 0$ for any S such that $|S| < m$, and $v(I) = c$, for some constant $c > 0$, where I denotes the grand bundle. We have that $v_2(S) = 0$ for every bundle S, and the maximum value $M(v)$ as in the statement of Theorem 3 equals $\frac{2c}{m(m+1)}$. Let $v' \in \mathbf{G_2}$ be the function that assigns weight $\frac{c}{m(m+1)}$ to every node in the levels 1 and 2 of H_I, and weight zero everywhere else. By Theorem 3, we have $|v(S) - v'(S)| \leq \frac{M(v)}{2} \cdot \frac{|S|(|S|+1)}{2}$ for any bundle S, i.e., $d(v, \mathbf{G_2}) \leq \frac{M(v)}{2} \cdot \frac{m(m+1)}{2}$. Furthermore, it is easy to see that the upper bound on $|v(S) - v'(S)|$ is matched for all bundles S, i.e., $|v(S) - v'(S)| = \frac{M(v)}{2} \cdot \frac{|S|(|S|+1)}{2}$ for any S. This implies that there does not exist a way of changing the weights on the nodes in the levels 1 and 2 of H_I that results in a smaller approximation error. To see this, assume the weight of, say, node u is augmented by $\epsilon > 0$, and let us denote with $\bar{v}$ the resulting valuation function. The error on the grand bundle will be reduced by ϵ. On the other hand, the error on every bundle S of cardinality $< m$ such that the sub-hypercube of dimension $|S|$ rooted at S includes u is increased by ϵ. Since

the approximation bound was matched on every possible bundle S, it follows that the overall approximation error of $\bar{v}$ will be increased by ϵ. It is easy to see that the same argument applies if the weight of any of the nodes in the levels 1 and 2 of H_I is reduced by an arbitrary amount. This implies the theorem.

*Proof (***Proof of Theorem 6.).** It is easy to see that (the decision variant of) the problem is in NP: for any assignment of items to bidders, we can compute the value of that assignment to each bidder in polynomial time, and sum these values to get the assignment's total value.

To show that the problem is NP-hard, we reduce an arbitrary instance of the NP-complete EXACT-COVER-BY-3-SETS problem to an instance of the winner determination problem as follows. Recall that in EXACT-COVER-BY-3-SETS, we are given a set S with $|S| = m$, and subsets $S_1, S_2, \ldots, S_n$ with $|S_j| = 3$ for all i, and are asked whether $\frac{m}{3}$ of the subsets cover S. Then, in our clearing problem, let there be an item i_s for every $s \in S$, and, for every $S_j = s_j^1, s_j^2, s_j^3$ (where s_j^1, s_j^2, s_j^3 is an arbitrary ordering of the elements of the subset), a bid b_{S_j} which places a value of 1 on edges $(i_{s_j^1}, i_{s_j^2})$ and $(i_{s_j^2}, i_{s_j^3})$, and places a value of 0 on everything else (including all vertices). We are asked whether it is possible to obtain a value of $\frac{2m}{3}$ in this auction. We show the instances are equivalent.

First suppose there exists an exact cover by 3-sets. Then, for each S_j in the cover, give the bidder corresponding to b_{S_j} the items $i_{s_j^1}, i_{s_j^2}, i_{s_j^3}$. This is a valid allocation because none of these sets of items overlap (because none of the sets in the cover overlap). Moreover, because each such bidder's value in this allocation is 2, and there are $\frac{m}{3}$ such bidders, the total value of the allocation is $\frac{2m}{3}$. So there exists an allocation that achieves the target value.

Now suppose there is an allocation that achieves the target value. Let $n(b)$ be the number of items allocated to the bidder corresponding to bid b, and let $v(b)$ be the value of the allocation to that bidder. Then the following must hold: if this bidder receives at least one item, we must have $\frac{v(b)}{n(b)} \leq \frac{2}{3}$. Moreover, the inequality is strict unless the bidder receives exactly the three items that are endpoints of his nonzero edges. The reason is the following: $v(b)$ can be at most 2, and will be less unless the bidder receives at least the three items that are endpoints of his nonzero edges, so this is certainly true for $n(b) \geq 3$. If $n(b) = 2$, then $v(b)$ can be at most 1 and the fraction can be at most $\frac{1}{2} < \frac{2}{3}$; if $n(b) = 1$, then $v(b) = 0$. Because the value of any allocation is $\sum_{b \in W} n(b) \frac{v(b)}{n(b)}$ (where W is the set of bids that win at least one item), it follows that the target value can be achieved if and only if all items are allocated to bidders, and $\frac{v(b)}{n(b)} = \frac{2}{3}$ for all $b \in W$. But because this equality holds only if every $b_{S_j} \in W$ receives items $i_{s_j^1}, i_{s_j^2}, i_{s_j^3}$, it follows that the S_j corresponding to winning bids in an allocation achieving the target value constitute an exact cover by 3-sets.

*Proof (***Proof of Theorem 7.).** The algorithm solves each tree in the forest separately. Fix a root r of the tree. For any vertex i in the tree, let $t(i,b)$ be the highest value that can be obtained in the auction from item i and its descendants alone (that is, if we throw away all other items), under the constraint that the bidder corresponding to bid b gets item i. Let $A(i,b)$ be the set of all allocations of the descendants that achieve this value. Then, for any clearing that assigns i to the bidder corresponding to b, without any loss we can change the allocation of the items in the subtree to be consistent with any element of $A(i,b)$ (assigning the descendants of i in the exact same manner); this will achieve at least as large total value from edges and vertices within the subtree; the value from all other vertices and all other edges that are disjoint from the subtree is clearly unaffected; and the only other edges that have one of the vertices of the subtree as an endpoint have i as that endpoint—and because i is still assigned to the bidder corresponding to b, they remain unaffected. Let $c_1, \ldots, c_{m_i}$ be the children of i. Then, we can conclude that $t(i,b) = v_b(i) + \sum_{k=1}^{m_i} \max\{v_b(i,c_k) + t(c_k,b), \max_{b' \neq b} t(c_k,b')\}$. This allows us to set up a simple dynamic program that will compute the $t(i,b)$ from the leaves upwards, and thus will eventually compute $t(r,b)$ for all b, and the highest such $t(r,b)$ is the optimal allocation value. We observe that for every bidder, for every edge (i,j), the value $v_b(i,j)$ is read exactly once; also, for any b and i, the expression $\max_{b' \neq b} t(i,b')$ takes only constant time to compute, because there are only two b''s for which we ever (for any b) need to look at $t(i,b')$: one that maximizes $t(i,b')$ (call it b_i1), and another one (b_i2) which maximizes $t(i,b')$ over all the remaining b' (which gives the second highest $t(i,b')$)—for the case where $b = b_i1$. It follows that the running time of the algorithm is $O(mn)$. The straightforward extension of the program to compute a best partial allocation $a(i,b)$ (with the restriction that i is allocated to the bidder corresponding to b) will also allow for computing an optimal allocation.

*Proof (***Proof of Theorem 8.).** The elicitor asks the valuation of any bundle of up to k items, i.e. $O(m^k)$ value queries. Then, the elicitor builds the G_k graph as follows: it first builds G_2 as described in Section 2. Let v_2 be the valuation function derived from G_2. For any three items bundle abc, the elicitor checks whether $v(abc) = v_2(abc)$. If not, a 3-edge between nodes a, b and c is inserted in the graph, with weight $v(abc) - v_2(abc)$. This way, graph G_3 is built, whose associated valuation function is denoted v_3. The process described above is repeated until G_k, and the associated valuation function v_k, is built. Since only up to k-wise dependencies occur in the original valuation function v, and G_k can express all such dependencies, it follows that $v_k = v$, and the theorem is proved.

*Proof (***Proof of Theorem 11.).** Let us consider an arbitrary valuation $v \in \mathbf{G}_{\mathbf{g(m)}}$, where $g(m)$ is an arbitrary function such that $g(m) \to \infty$ as $m \to \infty$. We observe that the problem of learning v can be equivalently restated as the problem of learning all the weights in $H_I(v)$. Since the auctioneer knows that the valuation is in $\mathbf{G}_{\mathbf{g(m)}}$, it can immediately conclude that $w(S) = 0$ for every bundle S such that $|S| > g(m)$. Furthermore, let us assume that the auctioneer

(somewhat magically) already knows the value of all the weights $w(S')$, for every bundle S' such that $|S'| < g(m)$. In other words, assume we are in the situation in which the only task left to the auctioneer is to learn the weights $w(\bar{S})$, where $|\bar{S}| = g(m)$. We claim that in the worst case at least $\binom{n}{g(m)}$ (which is super-polynomial in m) value queries are needed in order to accomplish this task.
In order to prove the claim, we first observe that the mere fact that the auctioneer knows the weights in levels $1, \ldots, g(m) - 1$ and $g(m) + 1, \ldots, m$ of the hypercube is not sufficient for it to infer the value of any of the weights at level $g(m)$. In fact, by the definition of weight, we have $w(\bar{S}) = v(\bar{S}) - \sum_{S' \subset \bar{S}} w(S')$, where $|\bar{S}| = g(m)$; thus, the mere knowledge of $\sum_{S' \subset \bar{S}} w(S')$ is not sufficient for the auctioneer to learn anything about $w(\bar{S})$. So, how could possibly the auctioneer increase its knowledge? We recall that the only tool for the auctioneer to increase its knowledge is by asking value queries. Observe that value queries regarding bundles S with $|S| < g(m)$ are useless, since the auctioneer already knows the value of all the weights in levels $1, \ldots, g(m) - 1$ of the hypercube. So, the auctioneer must ask queries on "large bundles". If the auctioneer ask the value of a certain bundle $\bar{S}$, with $|\bar{S}| = g(m)$, this information reveals the value of only one of the unkwown weights, $w(\bar{S})$. Since there are $\binom{n}{g(n)}$ such bundles $\bar{S}$, this approach would require asking a super-polynomial amount of value queries. So, the only possibility left for the auctioneer to reduce the communication burden is to ask the value of some bundle S, with $|S| > g(m)$. Assume the auctioneer asks the value of bundle S. Let us denote with $\bar{S}_1, \ldots, \bar{S}_c$ the subsets of cardinality $g(m)$ contained in S. We can write:

$$v(S) = w(S) + \sum_{i=1,\ldots,c} w(\bar{S}_i) + \sum_{S' \subset S, |S'| \neq g(m)} w(S') \ . \tag{1}$$

Since $v(S)$, $w(S)$ and $\sum_{S' \subset S, |S'| \neq g(m)} w(S')$ are known, (1) defines an equation with c unknowns (the weights of the bundles $\bar{S}_i$). We observe that:

a) the answers to different queries define different equations;
b) there are $\binom{m}{g(m)}$ unknowns in total;
c) all the equations are linearly independent.

Given *c*), a number of equations equal to the number of unknowns is necessary to learn the entire valuation function. Since there are super-polynomially many unknowns – *b*), and every value query defines a unique equation – *a*), it follows that super-polynomially many queries are needed to fully elicit the valuation. Note that even mixing value queries on bundles of cardinality $g(m)$ and queries on larger bundles would not change the situation, since the answer to a value query on a bundle S of cardinality $g(m)$ defines the unique simple equation $w(S) = v(S) - \sum_{S' \subset \bar{S}} w(S')$.

*Proof (***Proof of Theorem 12.).** It is straightforward that the valuations of individual items (1-wise dependencies) are uniquely defined. We proceed inductively, supposing that all j-wise dependencies for $1 \leq j \leq k$ are uniquely defined. Supposing for a moment that no larger dependencies were given, then for any

subset of size $k+1$, the j-wise dependencies for $1 \leq j \leq k$ define a valuation for this subset. If this is not the correct valuation, it can and must be adjusted with a $k+1$-wise dependency over these items that is exactly the difference between the correct valuation and the valuation implied by the j-wise dependencies for $1 \leq j \leq k$, because all other dependendencies that are still undefined do not concern a subset of these $k+1$ items.