New Lower Bound Techniques for the Query Complexity of Truthful Mechanisms

A survey of "Computational Efficiency Requires Simple Taxation" [Dob16]

Clay Thomas and Uma Girish

January 2019

Abstract

A major goal in algorithmic game theory is to understand the communication complexity of truthful combinatorial auctions. There were few known lower bounds on the number of value queries required, and no known lower bounds on the number of demand queries or amount of general communication required. This paper shows that for rich enough domains of valuation functions, the communication complexity of a truthful mechanism on that domain is characterized by the "taxation complexity" of that mechanism. A similar characterization is developed for the number of value queries and demand queries in terms of "menu complexity" and "affinity". This provides a new technique to lower bound the communication complexity of truthful mechanisms. The author also shows that truthful two-player mechanisms give rise to simultaneous mechanisms achieving the same welfare, suggesting the possibly easier approach of proving lower bounds for simultaneous communication problems.

1 Introduction

1.1 Preliminaries

A combinatorial auction is the problem of selling m items among n bidders, also called players. Each bidder i has a valuation function $v_i : 2^{[m]} \to \mathbb{R}_{\geq 0}$ on subsets of items. These valuations are monotone, so that the value for a bundle cannot decrease on adding more items. The seller wishes to communicate with the bidders and decide on the allocation. A mechanism for this problem is a complete specification of the interaction between the seller and the bidders, and an allocation rule. The allocation rule depends on this interaction and for each possible transcript, it specifies which bundle S_i to allocate to bidder i and at what price p_i. The seller decides upon a mechanism based on maximizing some property of the allocation; commonly studied ones include social welfare and revenue.

A fundamental question is to understand how robust mechanisms are to bidders behaving selfishly. Each bidder wants to maximize their own utility, which is $v_i(S_i) - p_i(S_i)$, where S_i is the set of items allocated to them. It might be possible for them communicate dishonestly with the bidder and end up with a better set. Truthfulness is a useful property for mechanisms to have, that make them robust to such behavior.

For "direct" mechanisms, i.e. those which ask a bidder to report their entire valuation function, the notion of "truthfulness" is simple: regardless of what other players do, you get higher utility by reporting your actual valuation function. For more general mechanisms, which interact with bidders in multiple rounds in arbitrary ways, things become more subtle. We define truthfulness as follows: any bidder i can (weakly) maximize their utility by answering queries according to their true valuation function v_i, assuming that other bidders answer in a way consistent with some valuation functions. Formally, this says that telling the truth is an *ex-post Nash equilibrium.* Informally, all it means is that we assume the strategies of the other players are not "crazy". There is a stronger notion, called "dominant strategy incentive compatible," which does not assume other bidders act in such a "non-crazy" way. In particular, in a dominant strategy incentive compatible mechanism, no matter what strategy is used by the other players, a bidder (weakly) maximizes his utility by telling the truth. We will not use this concept unless explicitly stated.

A truthful mechanism may alternately be viewed through the concept of a menu. Fix any setting v_{-i} of valuations of bidders other than i. Consider varying the valuation of player i and let $\mathcal{M}_{v_{-i}}$ be the collection of all possible sets and prices (S_i, p_i) that the player i would choose to buy. In other words,

$$\mathcal{M}_{v_{-i}} := \{\big(S_i(v_i, v_{-i}), p_i(v_i, v_{-i})\big) | v_i \in \mathcal{V}\}$$

where we define $S_i(v_i, v_{-i})$ and $p_i(v_i, v_{-i})$ as the set and price of the set allocated by the mechanism to player i when the valuations are $v_1, \ldots, v_n$ (note[1]). When the mechanism is truthful, this menu satisfies the following interesting property. If the player i has valuation v_i, then, among all the options on the menu $\mathcal{M}_{v_{-i}}$, the one corresponding to $S_i(v_i, v_{-i})$ maximizes his utility. That is, for every v_i', we have

$$v_i(S_i(v_i, v_{-i})) - p_i(v_i, v_{-i}) \geq v_i(S_i(v_i', v_{-i})) - p_i(v_i', v_{-i})$$

This is exactly the definition of truthfulness. This fact, that each player receives their favorite item on the menu, is known as the *taxation principle.* It allows us to interpret truthful mechanisms as implicitly presenting a menu to each player i based on the valuations v_{-i} of the other players and asking each player to choose their favorite option on their menus. Ignoring tie breaking issues for the moment, it turns out that any mechanism which does what we described is also truthful. This characterization essentially allows us to reason about menu mechanisms instead of arbitrary truthful mechanisms.

We further assume that menus are *monotonic*, that is, for every T, the price of any bundle containing T is at least the price of T. Dobzinski proves that this is actually without loss of generality, in the sense that any allocation function can be implemented with monotonic prices (and bidders will always be charged the same payments).

The communication complexity $cc(A)$ of a mechanism is the maximum number of bits exchanged between the bidders and the seller in any run of the mechanism. The value query complexity $val(A)$ of a mechanism is the maximum number of value queries made by the seller in any run of the mechanism, where the seller's communication is restricted to be value queries. The demand query complexity $dem(A)$ of a mechanism is the maximum number of demand queries made by the seller in any run of the mechanism, where the seller's communication is restricted to be demand queries. A major goal is to understand these complexities for truthful mechanisms and this paper presents a promising approach. We present an overview of the results below.

1.2 Statement Of Results

Since truthful mechanisms can thought of as implicitly presenting menus to players, if there are many possible menus, then intuitively, the players must communicate enough information to distinguish these menus. This leads Dobzinski to define the following quantity that captures the number

[1] Defining these functions S_i and p_i rigorously is a bit tricky. The mechanisms does not directly know the valuation functions v_i, so you must consider how the players will act before associating valuation functions with an outcome. For our purposes, they are the allocations and prices when players play honestly according to their valuation functions.

of possible menus.

$$tax(A) := \max_i \log \left|\{\mathcal{M}_{v_{-i}} | v_{-i} \in \mathcal{V}^{n-1}\}\right|$$

Dobzinski shows that $tax(A)$ characterizes the communication complexity $cc(A)$ of mechanisms A that are truthful for general valuations. That is,

$$tax(A) \quad \leq \quad cc(A) \quad \leq \quad poly(tax(A), \ldots)$$

Here, $(\ldots)$ stands for some technical terms which we'll ignore for the purposes of summary.

Dobzinski also develops a similar result for the value query complexity $val(A)$ and demand query complexity $dem(A)$. For value queries, the correct complexity measure is the maximal number of bundles a bidder might receive given any menu (the menu complexity $menu(A)$). For demand queries, a somewhat more subtle property (called the affinity $aff(A)$) of the prices given in the menus is needed. Dobzinski shows the following.

$$\begin{array}{rcccl} menu(A) & \leq & val(A) & \leq & poly(menu(A), \ldots) \\ aff(A) & \leq & dem(A) & & \end{array}$$

An important point is that in each of these query models, Dobzinski's complexity measure gives a lower bound for the amount of communication needed in that model. This leads us to new lower bound techniques for truthful mechanisms that we describe in the next section. In our project, we briefly review the left hand side of the first inequality (for a more detailed discussion, see the lecture notes), and give the details for the last two inequalities.

1.3 Application: Lower Bounds For truthful mechanisms

A major contribution of this work is to develop techniques to prove lower bounds on the communication complexity of (universally) truthful mechanisms. Previously, some lower bounds had been developed for restricted classes of valuation functions with value query access. It was known [DV16] that truthful mechanisms that achieve a $m^{1/2-\epsilon}$ approximation with value queries require exponentially many queries on submodular valuations. A similar result [Dug11] also existed for truthful in expectation mechanisms and for other query models of succinctly represented submodular valuations [DV12]. On the positive side, the most efficient truthful mechanism known is a $\sqrt{m}$ approximation due to Lavi and Swamy [LS05]. It is a major open question whether this is tight and this work provides a possible approach to settling this question.

The results of this paper show that mechanisms that are truthful for large enough domains must have communication/value query/demand query complexity to be at least the taxation/menu/affinity complexity respectively. This provides a technique to lower bound the former by studying the latter quantities. The only known application of this technique is the previous value query lower bound [DV16] which actually bounds the menu complexity.

1.4 Application: Simultaneous mechanisms from truthful mechanisms

This paper also shows that two-player truthful mechanisms give rise to (possibly not truthful) *simultaneous* mechanisms, where the two bidders simultaneously send a message to the seller who then decides the allocation. Furthermore, the communication complexity of the latter is at most the taxation complexity of the original mechanism. This gives rise to the simpler problem of lower bounding the communication complexity of a *simultaneous* mechanism.

Along this line, it is worth mentioning the work of Braverman, Mao and Weinberg [BMW18] that studies the communication complexity of the welfare estimation problem for arbitrary mechanisms on binary XOS valuations where such simulations were shown not to hold.

We now present some intuition for the construction of simultaneous mechanisms for two players. Here is an idea for a two round mechanism. Each player could present the other with the appropriate menu. The players could then choose their favorite option from the menu. A difficulty with this approach is that there may be multiple favorite options. For now, let us assume that in every menu and for every valuation, there is exactly one option that maximizes revenue, in which case our two round mechanism works. We can convert it into a simultaneous protocol as follows. Given menus $\mathcal{M}_1, \mathcal{M}_2$ for players 1 and 2, the seller decides to award to player 1, the union of all the best bundles that he could have wanted, conditioned on him producing the menu $\mathcal{M}_2$ for player 2. More precisely, he gives $S_1 = \cup_{v_1 \text{ produces } \mathcal{M}_2} BestOption(v_1, \mathcal{M}_1)$ to player 1 and the rest to player 2. The reason this works is that conditioned on producing $\mathcal{M}_1, \mathcal{M}_2$, any best option for player 2 must be disjoint from every best option for player 1, otherwise the best options for player 1 and player 2 would conflict.

The argument in the general case where there are multiple favorite options is a perturbation argument. Dobzinski argues that it is reasonable to assume that any truthful mechanism can 'handle' small (but not too small) perturbations and a small perturbation has the effect of making the utility maximizing bundle unique.

2 Review: Arbitrary Communication & Taxation Complexity

This section gives a very brief account of the notion of taxation complexity, and why it gives a lower bound on communication complexity. For more details, see the lecture notes, and for the full details, see Dobzinski's paper.

Definition. *Let A be a mechanism, truthful over a class of valuations $\mathcal{V}$. The* taxation complexity *$tax(A)$ of A is the logarithm of the maximum number of distinct menus which might be presented to a player i:*

$$tax(A) = \max_i \log |\{\mathcal{M}_{v_{-i}} | v_{-i} \in \mathcal{V}^{n-1}\}|$$

This next proof introduces a key trick which all of our lower bound proofs will use: the idea of treating the menu itself as a valuation function, and passing that as an input to the mechanism. What we will show is that, given any distinct menus, we can cleverly come up with instances which the mechanism needs more communication (i.e. another transcript) in order to tell apart.

Theorem 1. *Let A be a deterministic mechanism which is truthful for general monotonic valuation functions. Then $tax(A) \leq cc(A) + 1$.*

Proof. First, assume that the final bit of the transcript of A is 1 if player i gets positive utility, and 0 otherwise. We can do this (while maintaining truthfulness) by simply asking bidder i whether he gets positive utility as the last step of the mechanism, adding one bit of communication. This seems like a strange thing to do, but the reason is the following: we need to be able to tell from the transcript alone whether player i gets positive utility.

Now, consider any menu $\mathcal{M}$ presented to player i, and let v_{-i} denote the valuations functions for bidders other than i which present menu $\mathcal{M}$ to i. Let $v_{\mathcal{M}}$ be the valuation function such that $v_{\mathcal{M}}(S) = \mathcal{M}(S)$ if $\mathcal{M}(S) < \infty$, and $v_{\mathcal{M}}(S) = B$ otherwise, where B is strictly larger than any finite price in $\mathcal{M}$. Note that, because we assume the menus $\mathcal{M}$ are monotonic, $v_{\mathcal{M}}$ is monotonic. A crucial fact is the following: when mechanism A is run on input $(v_{\mathcal{M}}, v_{-i})$, bidder i gets utility zero, because i has value exactly equal to the price he needs to pay to receive any particular bundle.

Let $\mathcal{M}' \neq \mathcal{M}$ be a distinct menu presented by different valuation functions v'_{-i}. The crux of the proof is the following claim:

Claim 2. *The mechanism A cannot have the same transcript on inputs $(v_{\mathcal{M}}, v_{-i})$ and on $(v_{\mathcal{M}'}, v'_{-i})$.*

Proof. Suppose for contradiction that the transcript of A was identical on $(v_{\mathcal{M}}, v_{-i})$ and on $(v_{\mathcal{M}'}, v'_{-i})$. Call this transcript T. By our assumptions, T must end in zero because bidder i gets zero utility. The menus are distinct, so one of them must give a strictly greater price for some bundle S. Without loss of generality suppose $\mathcal{M}(S) > \mathcal{M}'(S)$. A standard rectangle argument from communication complexity tells us that mechanism A must also use transcript T on input $(v_{\mathcal{M}}, v'_{-i})$.

Now, when other bidders have values v'_{-i}, bidder i is presented with menu $\mathcal{M}'$. *Because the mechanism A is truthful*, bidder i must get his utility-maximizing bundle from $\mathcal{M}'$. Observe that, in menu $\mathcal{M}'$, bidder i with valuation $v_{\mathcal{M}}$ gets positive utility from set S. Thus, bidder i must get positive utility on input $(v_{\mathcal{M}}, v'_{-i})$. However, the mechanism A has transcript T on this input, and T ends in zero (corresponding to bidder i getting utility zero). This is a contradiction, proving the claim. $\square$

Thus, there are at least as many transcripts as there are menus, specifically:

$$2^{tax(A)} \leq \text{(number of menus for } i\text{)} \leq \text{(number of transcripts of } A\text{)} \leq 2^{cc(A)}$$

Thus $tax(A) \leq cc(A)$. The extra $+1$ in the theorem statement comes from the assumption on the final bit sent by A. $\square$

3 Value Queries & Menu Complexity

In this section we establish the equivalence between value query complexity $val(A)$ and the menu complexity $menu(A)$ of mechanisms A that are truthful on all valuations.

3.1 Lower Bound

Definition. *We say a bundle S is* strictly in *a menu $\mathcal{M}$ if, for each $T \supset S$, $T \neq S$, we get $\mathcal{M}(T) > \mathcal{M}(S)$. Also, the grand bundle M is in the menu if it has finite price: $\mathcal{M}(M) < \infty$.*

The menu complexity *of a mechanism A is then*

$$menu(A) = \max_i \max_{v_{-i}} |\{S | S \textit{ is strictly in } M_{v_{-i}}\}|$$

The proof will use a similar high-level idea as the one for general communication. This time, we fix a menu, and consider different bundles in that menu. Using some cleverness (using the menu itself as input) we provide inputs for which the mechanism needs more communication (i.e. needs to make some specific value queries) in order to tell apart.

Theorem 3. *Let A be a deterministic mechanism which is truthful for general monotonic valuation functions and which only uses value queries. Then $menu(A) \leq val(A) + 2$.*

Proof. Let $\mathcal{M}$ be an arbitrary menu presented to player i by valuations v_{-i} of the other players. Let $v_{\mathcal{M}}$ be the valuation function such that $v_{\mathcal{M}}(S) = \mathcal{M}(S)$ if $\mathcal{M}(S) < \infty$ and $v_{\mathcal{M}}(S) = B$ otherwise, where B is strictly larger than any finite price in $\mathcal{M}$. Observe that player i gets utility zero on input $(v_{\mathcal{M}}, v_{-i})$. Let S_0 be the bundle awarded to player i on input $(v_{\mathcal{M}}, v_{-i})$.

Now, let S be any bundle strictly in $\mathcal{M}$ and let $S \neq \emptyset$ and $S \neq S_0$. Define a new input instance via the valuation v_S such that $v_S(S) = v_{\mathcal{M}}(S) + \epsilon$, and $v_S(T) = v_{\mathcal{M}}(T)$ for $T \neq S$, where $\epsilon > 0$ is small enough that v_S is still monotonic. Such an ϵ exists because S is strictly in $\mathcal{M}$.

Consider the execution of A on input $(v_{\mathcal{M}}, v_{-i})$. We'll show that, because A is truthful, it must query the value of S for every $S \neq S_0, \emptyset$. Suppose for contradiction that A does not query bidder i on the set S. Then it will allocate to player i the same bundle on input (v_S, v_{-i}) as it would on $(v_{\mathcal{M}}, v_{-i})$, namely S_0, because the value queries for both inputs will be exactly the same. This means player i will get utility zero. However, $v_S(S) - \mathcal{M}(S) = \epsilon > 0$, i.e. player i has a menu option that gets him positive utility. Because the mechanism is truthful, player i should get his favorite option on the menu. This is a contradiction.

Thus, the mechanism must query every set strictly in the menu $\mathcal{M}$, other than possibly $\emptyset$ and S_0. Thus, $val(A) \geq menu(A) - 2$, as desired. $\square$

3.2 Upper Bound

In this section we prove that given any truthful mechanism A, there is a mechanism A' which simulates A with value query complexity at most $poly(menu(A), \ldots)$. We remark that there is a step in the proof that isn't true as stated, and we are unable to fix this. We will indicate this as it appears.

A naive implementation is the following. For each i, the seller communicates with the players $\neq i$ and outputs the menu $\mathcal{M}_{v_{-i}}$ for the player i. Each player then selects a utility maximizing bundle in their menu and the seller awards them this option. This implementation is incorrect because of tie-breaking: There may be several options that maximize a bidder's utility and the seller must award items consistently across bidders, without selling an item to more than one bidder. This prompts us to define the following quantity.

Definition. *$tie^{val}(A)$ is the communication complexity of the following $n+1$-player problem. Each player i has a valuation function v_i and a menu $\mathcal{M}_i$ from which they must pick their favorite option. We assume that everyone knows each $\mathcal{M}_i$. The seller wishes to interact with the bidders and find an allocation of items so that each player gets an option of maximum utility from their menu.*

Another difficulty with this implementation is in efficiently finding and computing the menus $\mathcal{M}_{v_{-i}}$. It is not even clear how to compactly represent this menu. This leads us to define the following.

Definition. *$price^{val}(A)$ is the communication complexity of the following n-player problem. The players $\neq i$ have valuations v_{-i} and the seller is given an input bundle $S \subseteq [m]$. The seller wishes to interact with the bidders and find out the price $\mathcal{M}_{v_{-i}}(S)$ of that bundle in the menu presented to player i by the other players.*

The idea is to show that that menu for player i can be explicitly constructed by players $\{j : j \neq i\}$ with $poly(price^{val}(A), menu(A))$ value queries. The final mechanism is then as before. For each i, the seller communicates with players $\neq i$ to find the menu that player i should get, and presents it to them (in a compact way). The seller further communicates with the players using $tie^{val}(A)$ value queries to break ties and decide on the allocation. It thus suffices to show that the menus can be computed and represented efficiently.

Let us first see how to compactly represent the menu for player i. Note that for every set S, every bundle containing it has price at least $\mathcal{M}(S)$, as we assume menus are monotonic. Furthermore, if S is not strictly in the menu, this means that there is some bundle $T = S \cup \{j\}$ whose price is same as that of S. Either T is strictly in the menu or we can repeat this procedure to find some strictly larger bundle whose price is the same. This process must end at a bundle strictly in the menu. This shows us that for every S, there is some bundle containing S that has price exactly

$\mathcal{M}(S)$ and is strictly in the menu. Thus, the price function is determined by its values on those bundles strictly in the menu, specifically

$$\mathcal{M}(S) = \min\{\mathcal{M}(T) | T \supseteq S, T \text{ is strictly in } \mathcal{M}\}$$

Since there are at most $menu(A)$ bundles strictly in the menu, the price function can be described by the prices at these bundles. We now show how to efficiently construct the menu for $i = n$ and the others follow by symmetry.

Theorem 4 (Menu Reconstruction Theorem)**.** *The menu for player n can be explicitly constructed by players $\{1, \ldots, n-1\}$ with $poly(price^{val}(A), menu(A), m)$ value queries.*

Proof. Recall that $price^{val}(A)$ captures the value query complexity of deciding the price $\mathcal{M}_{v_{-n}}(S)$ of a bundle S in the menu $\mathcal{M}_{v_{-n}}$ presented to n by the players $\{1, \ldots, n-1\}$. The intuition for the reconstruction theorem is that there are at most $menu(A)$ bundles strictly in any menu, so if the players knew which bundles those were, they could find their using $price^{val}(A)$ queries per bundle and thus infer the prices of every set. We now present an efficient protocol for learning these bundles.

Claim 5. *Let $0 = p_0 < p_1 < \ldots < p_k$ be the distinct prices that appear in the menu $\mathcal{M}_{v_{-n}}$. (The price of the empty set is set to 0). Let N_i be the number of bundles strictly in $\mathcal{M}_{v_{-n}}$ with price at most p_i. Then, for each i, there is a communication protocol for players $[n-1]$ with $poly(N_i, price^{val(A)}, m)$ value queries that outputs the bundles of price at most p_i which are strictly in $\mathcal{M}_{v_{-n}}$. Furthermore, any such protocol necessarily queries some set of price p_{i+1}.*

The theorem follows from this claim by an inductive argument. We run the above algorithm for $i = 0$ to obtain all bundles of price at most $p_0 = 0$. While doing this, we would have queried some set of price p_1. We can thus find p_1 by computing the minimum price of all sets queried whose price is not p_0. We thus obtain p_1 and repeat this process to find bundles of price at most p_2 and so on until we reach the bundle of largest price p_k. The required bound follows because the total number of value queries is at most

$$\sum_i poly(N_i, price^{val(A)}, m) \leq poly\Big(\sum_i N_i, price^{val}(A), m\Big) \leq poly(menu(A), price^{val}(A), m)$$

The last line follows because of the following inequality bounding $\sum_i N_i$.

$$\sum_i N_i = (k+1)N_0 + \sum_{i=0}^{k-1}(k-i)(N_{i+1} - N_i) \leq menu(A)\big(N_0 + \sum_{i=0}^{k} N_{i+1} - N_i\big) \leq O(menu(A)^2)$$

This shows us how to use claim 4 to prove the reconstruction theorem. We now prove claim 4.

Proof of claim 5. We first prove that any algorithm that finds the bundles strictly in the menu $\mathcal{M}$ of price at most p_i must query some bundle of price p_{i+1}. Suppose not, consider the menu $\mathcal{M}'$ obtained by altering $\mathcal{M}$ as follows. For each set of price p_{i+1}, change its price to p_i. Note that there is at least one bundle of price p_i strictly in $\mathcal{M}'$ that is not strictly in $\mathcal{M}$. This gives us a contradiction to the assumption that the algorithm outputs all bundles of price at most p_i strictly in the menu.

We now present Dobzinski's algorithm to find the bundles. To simplify notation, Dobzinski introduces the valuation function v which is 0 on bundles whose price is at most p_i and 1 on bundles whose price is more than p_{i+1}. Let $\mathcal{A} = \{A_1, \ldots, A_{N_i}\}$ be the set of bundles of price p_i

which are strictly in the menu, i.e. those for which $v(A_i) = 0$ and $v(T) = 1$ for every strict superset T of A_i. We wish to find out $A_1, \dots, A_{N_i}$. Note that $v(S) = 0$ precisely when $S \subseteq A_j$. This suggests the following recursive algorithm for finding $\mathcal{A}$. We define Find(S, *Allowed*) to be the set of bundles strictly in $\mathcal{M}$ of price p_i which are of the form $S \cup T$ for $T \subseteq$ *Allowed*. Note that Find($\emptyset, M$) is $\mathcal{A}$. The following is a recursive procedure for Find(S, *Allowed*).

```
1: function FIND($S$, $Allowed$)
2:     Compute $\mathcal{M}(S)$
3:     if $v(S) = 1$ then return $\emptyset$
4:     Compute $v(S \cup \{j\})$ for every $j \in [m]$
5:     if all of these are 1 then return $\{S\}$
6:     Let $\mathcal{A} = \emptyset$
7:     for every item $j \in Allowed$ do
8:         Remove the smallest element $j$ from $Allowed$
9:         Add the elements of Find($S \cup \{j\}, Allowed$) to $\mathcal{A}$
       return $\mathcal{A}$
```

Note that the recursive call to Find in line 9 computes all bundles that contain S, contain no element of $[j-1]$, contain j and whose other elements are in $\{j+1, \dots, n\}$. Induction can now be used to argue the correctness of this procedure. It remains to show that the number of value queries this protocol makes is $poly(N_i, price^{val}(A), m)$.

Consider running Find($\emptyset, M$), and construct a tree whose nodes are labeled by all those bundles S for which Find(S, *Allowed*) is called for some set *Allowed*. The tree is rooted at $\emptyset$ and the children of a node S are those of the form $S \cup \{j\}$ for some $j \in M$. Note that a set S is passed as input to Find($\circ, \circ$) at most once, because we remove each j from *Allowed* whenever j is added to S. So, counting the number of nodes in this tree will help us bound the number of recursive calls to Find(S, *Allowed*) which will in turn help us bound the number of value queries. We count the number of nodes by counting the number of leaves. The leaves S of this tree satisfy either:

(1.) $v(S) = 0$, yet $v(S \cup \{j\}) = 1$ for each j. In this case, S is a bundle strictly in $\mathcal{M}$, i.e. $S \in \mathcal{A}$.

(2.) $v(S) = 1$. In this case, S must be of the form $S' \cup \{j\}$, where $v(S') = 0$. Now, Dobzinski claims that S' must be contained in some path from the root to a set in $\mathcal{A}$. The idea behind this is that the parent of S, being of price at most p_i must be contained in some bundle T of price p_i strictly in the menu and this set T is what we want. The problem is that this bundle T need not be a child of S' in the tree defined by the recursion, since it may have been explored earlier and the set *Allowed* may dictate that T cannot be a child of S'.

Because (2.) is not quite true, it turns out that the algorithm as stated can produce a tree with $\approx 2^n$ leaves when run on an instance where $v([m-1]) = 1, v([m]) = 1$ and 0 everywhere else. For completeness, we continue presenting Dobzinski's proof assuming (2.). Each path from the root to a set in $\mathcal{A}$ has length at most m. So there are at most mN_i nodes along such paths. Via the case analysis above, every call to Find corresponds to a node along such a path, or a child of a node along such a path. Thus, there are at most $m^2 N_i$ calls to Find. Ignoring recursion, each call to Find valuates v at most $m+1$ times, and each evaluation uses at most $price^{val}(A)$ value queries. Thus, running Find($\emptyset, M$) uses at most

$$(m+1)m^2 N_i price^{val}(A) = poly(m, N_i, price^{val}(A))$$

value queries, as desired.

□

□

3.3 Truthfulness of the new Mechanism

In this section, we explore the truthfulness of the mechanism described above. Recall that the mechanism is to have the menu of each player i presented by the others. The players then choose their favorite options and the seller allocates items using the tie-breaking algorithm.

Firstly, telling the truth is an ex-post nash equillibrium. That is, provided the players $\neq i$ honestly present the menus to each player, it is preferable for player i to play honestly as well. This can be seen as follows. Suppose the menus were honestly presented, then the players needn't lie about their most favorite option on the menu, since that is what they get anyway. Furthermore, the players might as well tie break honestly, since their utilities for tied sets are the same. The only other place a player could lie is while deciding the menu for other players. The menu that any player i gets is guaranteed to be the correct menu $\mathcal{M}_{v_{-i}}$ because the other players are playing honestly. Since he can only get something on the menu $\mathcal{M}_{v_{-i}}$ and this menu is fixed by the other players regardless of what he does, he might as well tell the truth while constructing the menu for other players.

It appears that player i's menu is independent of what he does, and hence this mechanism should be DSIC as well. This need not be true. For instance, if the strategy of players $2, \ldots, n$ is to present a non trivial menu to player 1 only if he reports his valuation as v_0, then player 1 would be better off by reporting his valuation as v_0, regardless of what his true valuation is. Dobzinski points out that the existence of such a mechanism is an open question for more than three players. That is, an efficient conversion of ex-post truthful mechanisms to DSIC mechanisms using menus is open for $n \geq 3$. He presents a modification of the previously mentioned algorithm that works for two players.

4 Demand Queries

Mechanisms using demand queries are notoriously hard to reason about, indeed, *no* lower bounds for truthful demand-query mechanisms are known. However, Dobzinski thinks the following definition and theorem may be a promising way to approach such lower bounds. The definitions may seem a bit counter-intuitive or complex, but it will turn out to interact with the definition of a demand query in just the right way.

Definition. *A menu $\mathcal{M}$ is called α-min-affine if there exist price vectors $p^{(i)}$ (with each price $p_j^{(i)} \in \mathbb{R}_{\geq 0} \cup \{\infty\}$) and numbers $r^{(i)}$ for $i = 1, \ldots, \alpha$, where*

$$\mathcal{M}(S) = \min_i \left(r^{(i)} + \sum_{j \in S} p_j^{(i)} \right)$$

The affinity *$aff(A)$ of a mechanism A is the minimal α such that all menus presented by A are α-min-affine.*

When we reasoned about value queries, we saw that the bundles strictly in a menu $\mathcal{M}$ determined which bundles needed to be queried by the mechanism. Our reasoning for demand queries will go in the other direction: we'll show that the demand queries made to a bidder i will determine the affinity of the menu that is actually presented to the bidder i. Thus, if some menu requires high affinity to represent, the demand query complexity of the mechanism will be high.

Theorem 6. *Let A be a deterministic mechanism which is truthful for general valuations which uses only demand queries. If A makes at most α demand queries, then every menu presented by A is α-min-affine. In particular, $aff(A) \leq dem(A)$.*

Proof. Suppose menu $\mathcal{M}$ is presented to player i on inputs v_{-i}. Let $v_{\mathcal{M}}$ be the valuation function such that $v_{\mathcal{M}}(S) = \mathcal{M}(S)$ if $\mathcal{M}(S) < \infty$ and $v_{\mathcal{M}}(S) = (m+1)B$ otherwise, where B is strictly larger than any finite price in $\mathcal{M}$. The proof proceeds by showing that, when the mechanism A is run on input $(v_{\mathcal{M}}, v_{-i})$, the price vectors on which the demand queries are made must describe the menu, as specified in the definition of affinity.

Specifically, run the mechanism A on input $(v_{\mathcal{M}}, v_{-i})$, and denote the price vector of the kth demand query made to player i by $p^{(k)}$. Let $D^{(k)}$ be the set returned by player i, i.e. i's utility maximizing bundle. Define

$$r^{(k)} = v_{\mathcal{M}}(D^{(k)}) - \sum_{j \in D^{(k)}} p_j^{(k)}$$

That is, $r^{(k)}$ gives the profit of player i on the kth demand query. Our goal will be to describe the menu $\mathcal{M}$ in terms of the price vectors $p^{(k)}$ and profits $r^{(k)}$, as in the definition of affinity $\mathcal{M}$. Observe that the they already do this perfectly for $D^{(k)}$, i.e. $\mathcal{M}(D^{(k)}) = r^{(k)} + \sum_{j \in D^{(k)}} p_j^{(k)}$. Intuitively, what we would like to show is that we can interpolate this fact between the different sets $D^{(k)}$, using the fact that demand queries return the utility maximizing bundle. This is what the next two claims do.

Claim 7. *If $\mathcal{M}(S) < \infty$, then for all k we have $\mathcal{M}(S) \le r^{(k)} + \sum_{j \in S} p_j^{(k)}$.*

Proof. Because $D^{(k)}$ is the profit-maximizing bundle on prices $p^{(k)}$, we have

$$r^{(k)} = v_{\mathcal{M}}(D^{(k)}) - \sum_{j \in D^{(k)}} p_j^{(k)} \ge v_{\mathcal{M}}(S) - \sum_{j \in S} p_j^{(k)} = \mathcal{M}(S) - \sum_{j \in S} p_j^{(k)}$$

□

Claim 8. *If $\mathcal{M}(S) < \infty$, then there exists a k for which $\mathcal{M}(S) \ge r^{(k)} + \sum_{j \in S} p_j^{(k)}$*

Proof. **Case 1:** First, suppose S is strictly in the menu $\mathcal{M}$, that is, for each strict superset T of S, we have $\mathcal{M}(T) > \mathcal{M}(S)$. For $\epsilon > 0$, define a new valuation function v_ϵ such that $v_\epsilon(T) = v_{\mathcal{M}}(T)$ if $T \neq S$, and $v_\epsilon(S) = v_{\mathcal{M}}(S) + \epsilon$. Consider all $\epsilon > 0$ such that v_ϵ remains monotone.

Observe that, by the definition of $v_{\mathcal{M}}$, player i will certainly get utility zero when his valuation function is $v_{\mathcal{M}}$. However, if i's input were v_ϵ, he could get utility exactly ϵ by receiving bundle S. Because the mechanism A is truthful, this means it must allocate S to player i in this case. Thus, the execution of the mechanism on input (v_ϵ, v_{-i}) cannot be the same as on input $(v_{\mathcal{M}}, v_{-i})$, specifically, some demand query must return a different bundle from $D^{(k)}$. Let k_ϵ denote the smallest k for which the demand queries deviate, i.e. a bundle other than $D^{(k)}$ is returned. Because no valuation other than that of S has changed, observe that this demand query must return S, and we have $v_\epsilon(S) - \sum_{j \in S} p_j^{(k_\epsilon)} \ge r^{(k_\epsilon)}$.

Now, k_ϵ can only increase as $\epsilon \to 0$. Indeed, if $\delta > \epsilon$, we get $v_\delta(S) - \sum_{j \in S} p_j^{(k_\epsilon)} > v_\epsilon(S) - \sum_{j \in S} p_j^{(k_\epsilon)} \ge r^{(k_\epsilon)}$, so the demand queries made on v_δ will deviate from those on $v_{\mathcal{M}}$ no later than k_ϵ. Now, because the number of demand queries α is finite, we can take $k = \lim_{\epsilon \to 0} k_\epsilon$, and let $\epsilon_0 > 0$ be such that whenever $\epsilon < \epsilon_0$, we get $k_\epsilon = k$. This tells us exactly that

$$v_{\mathcal{M}}(S) + \epsilon = v_\epsilon(S) > r^{(k)} + \sum_{j \in S} p_j^{(k)}$$

for all $0 < \epsilon < \epsilon_0$. Thus, $\mathcal{M}(S) = v_{\mathcal{M}}(S) \ge r^{(k)} + \sum_{j \in S} p_j^{(k)}$, as desired.

Case 2: Now, we return to the second case, where S is not strictly in the menu, i.e. $\mathcal{M}(S) = \mathcal{M}(T')$ for some T' a strict superset of S. (Essentially we will show that the assumption that S was strictly in the menu was without loss of generality). Let T be a maximal such T. Because T is maximal, T is strictly in the menu $\mathcal{M}$. Thus, case 1 applies to T. So let k be such that $v_{\mathcal{M}}(T) \geq r^{(k)} + \sum_{j \in T} p_j^{(k)}$. Indeed, this immediately implies that

$$\mathcal{M}(S) = v_{\mathcal{M}}(S) = v_{\mathcal{M}}(T) \geq r^{(k)} + \sum_{j \in T} p_j^{(k)} \geq r^{(k)} + \sum_{j \in S} p_j^{(k)}$$

□

We also need to handle the case where the bundle S cannot be acquired in the menu, i.e. $\mathcal{M}(S) = \infty$. This lemma also explains why we defined $v_{\mathcal{M}}(S)$ the way we did in cases where $\mathcal{M}(S) = \infty$.

Claim 9. *If $\mathcal{M}(S) = \infty$, then $r^{(k)} + \sum_{j \in S} p_j^{(k)} \geq (m+1)B$ for every k.*

Proof. Suppose $\mathcal{M}(S) = \infty$. We get $r^{(k)} \geq v_{\mathcal{M}}(S) - \sum_{j \in S} p_j^{(k)} = (m+1)B - \sum_{j \in S} p_j^{(k)}$, and rearranging proves this direction.

Now, suppose $\mathcal{M}(S) < \infty$. □

Now time to fix some technicalities and wrap up.

$$\widetilde{p}_j^{(k)} = \begin{cases} \infty & \text{if } p_j^{(k)} \geq B \text{ or } r^{(k)} \geq B \\ p_j^{(k)} & \text{otherwise} \end{cases}$$

Let $\mathcal{M}'$ be the min-affine menu described by the prices $\widetilde{p}^{(k)}$ and the numbers $r^{(i)}$, that is, $\mathcal{M}'(S) = \min_k \left(r^{(k)} + \sum_{j \in S} \widetilde{p}_j^{(k)} \right)$. We'll show that $\mathcal{M} = \mathcal{M}'$. We need two simple cases:

Suppose $\mathcal{M}(S) < \infty$. By the first two claims there exists a k such that $\mathcal{M}(S) = r^{(k)} + \sum_{j \in S} p_j^{(k)}$. Because $\mathcal{M}(S) < B$, in particular each $p_j^{(k)} < B$, so $\widetilde{p}_j^{(k)} = p_j^{(k)}$. Observe also that we always have $\widetilde{p}_j^{(k')} \geq p_j^{(k')}$. Thus, $\mathcal{M}(S) = \mathcal{M}'(S)$, as desired.

Now suppose $\mathcal{M}(S) = \infty$. Consider any k. By the final claim, $r^{(k)} + \sum_{j \in S} p_j^{(k)} \geq (m+1)B$. Now, the left hand side of this is a sum of at most $m+1$ nonnegative numbers, so at least one of these numbers must be at least B. If there is some j where $p_j^{(k)} \geq B$, then $\widetilde{p}_j^{(k)} = \infty$, so $r^{(k)} + \sum_{j \in S} \widetilde{p}_j^{(k)} = \infty$. If $r^{(k)} \geq B$, then the previous sentence holds for every j. Thus, we have $r^{(k)} + \sum_{j \in S} \widetilde{p}_j^{(k)} = \infty$ for every k, so $\mathcal{M}'(S) = \infty$, as desired. □

4.1 Upper Bound

We mention that no menu reconstruction theorem exists for demand queries. That is, there are mechanisms that make polynomially many demand queries to find the price and the allocation, but to reconstruct the menu requires exponentially many demand queries. The example of this given in Dobzinski's full paper is actually fairly simple, but the analysis is somewhat complex. See the paper for details.

References

[BMW18] Mark Braverman, Jieming Mao, and S. Matthew Weinberg. “On Simultaneous Two-player Combinatorial Auctions”. In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms.* SODA ’18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, pp. 2256–2273. ISBN: 978-1-6119-7503-1. URL: `http://dl.acm.org/citation.cfm?id=3174304.3175451`.

[Dob16] Shahar Dobzinski. “Computational Efficiency Requires Simple Taxation”. In: *CoRR* abs/1604.01971 (2016). arXiv: `1604.01971`. URL: `http://arxiv.org/abs/1604.01971`.

[Dug11] Shaddin Dughmi. “Limitations of randomized mechanisms for combinatorial auctions”. In: *In Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS*. 2011.

[DV12] Shahar Dobzinski and Jan Vondrak. “The Computational Complexity of Truthfulness in Combinatorial Auctions”. In: *Proceedings of the 13th ACM Conference on Electronic Commerce.* EC ’12. Valencia, Spain: ACM, 2012, pp. 405–422. ISBN: 978-1-4503-1415-2. DOI: `10.1145/2229012.2229044`. URL: `http://doi.acm.org/10.1145/2229012.2229044`.

[DV16] Shahar Dobzinski and Jan Vondrák. “Impossibility Results for Truthful Combinatorial Auctions with Submodular Valuations”. In: *J. ACM* 63.1 (Feb. 2016), 5:1–5:19. ISSN: 0004-5411. DOI: `10.1145/2786754`. URL: `http://doi.acm.org/10.1145/2786754`.

[LS05] Ron Lavi and Chaitanya Swamy. “Truthful and near-optimal mechanism design via linear programming”. In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS’05)* (2005), pp. 595–604.