# On the (in)-approximability of Bayesian Revenue Maximization for a Combinatorial Buyer

NATALIE COLLINA*, Princeton University
S. MATTHEW WEINBERG†, Princeton University

We consider a revenue-maximizing single seller with $m$ items for sale to a single buyer whose value $v(\cdot)$ for the items is drawn from a known distribution $D$ of support $k$. A series of works by Cai et al. establishes that when each $v(\cdot)$ in the support of $D$ is additive or unit-demand (or $c$-demand), the revenue-optimal auction can be found in $\text{poly}(m, k)$ time.

We show that going barely beyond this, even to matroid-based valuations (a proper subset of Gross Substitutes), results in strong hardness of approximation. Specifically, even on instances with $m$ items and $k \leq m$ valuations in the support of $D$, it is not possible to achieve a $1/m^{1-\varepsilon}$-approximation for any $\varepsilon > 0$ to the revenue-optimal mechanism for matroid-based valuations in (randomized) poly-time unless NP $\subseteq$ RP (note that a $1/k$-approximation is trivial).

Cai et al.'s main technical contribution is a black-box reduction from revenue maximization for valuations in class $\mathcal{V}$ to optimizing the difference between two values in class $\mathcal{V}$. Our main technical contribution is a black-box reduction in the other direction (for a wide class of valuation classes), establishing that their reduction is essentially tight.

CCS Concepts: • **Theory of computation** → **Algorithmic mechanism design**.

Additional Key Words and Phrases: optimal mechanism design; revenue; reductions; gross substitutes.

**ACM Reference Format:**
Natalie Collina and S. Matthew Weinberg. 2020. On the (in)-approximability of Bayesian Revenue Maximization for a Combinatorial Buyer. In *Proceedings of the 21st ACM Conference on Economics and Computation (EC '20), July 13–17, 2020, Virtual Event, Hungary.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3391403.3399496

## 1 INTRODUCTION

"Multi-dimensional mechanism design" has been a driving force in Mathematical Economics since Myerson's seminal work [29], and also in Algorithmic Game Theory since its introduction to TCS by seminal work of Chawla, Hartline, and Kleinberg [12]. The problem has gained broad interest within TCS owing to the complexity of optimal solutions [5, 16–20, 25, 26, 30–32]. In light of this, there now exists a substantial body of work providing algorithms to find optimal (or approximately optimal) auctions, accepting that the resulting solution may be randomized, non-monotone, or not particularly simple [1–3, 6–9, 21, 22].

To properly parse the results of our work, we need to be clear about the input model and objective. The seller has $m$ items for sale to a single buyer, and is given as input an explicit finite-support

*nataliecollina@gmail.com.
†smweinberg@princeton.edu. Supported by NSF CCF-1717899.


*EC '20, July 13–17, 2020, Virtual Event, Hungary*

ACM ISBN 978-1-4503-7975-5/20/07…$15.00
https://doi.org/10.1145/3391403.3399496

distribution $D$ over valuation functions in some class $\mathcal{V}$ (given by listing the valuations along with the probability with which they are drawn from $D$).[1] The designer's goal is to output a description of an auction which (approximately) maximizes expected revenue. The main positive results of these prior works provide poly-time algorithms for the revenue-optimal *multi-buyer* auction, when all buyers are additive/unit-demand/$c$-demand (which of course imply poly-time algorithms for a single buyer as well, although the single-buyer case does not require many of the developed tools).[2] [9] also establishes a hardness result: there exists a constant $d$ such that it is not possible to guarantee a $1/m^d$-approximation for a single buyer whose valuation is submodular[3] in poly-time unless NP $\subseteq$ RP. Our first main result establishes that revenue-maximization is inapproximable immediately beyond $c$-demand valuations:

**Informal Theorem 1** (See Theorem 13). *Unless NP $\subseteq$ RP, for all $\varepsilon > 0$, there is no poly-time $1/m^{1-\varepsilon}$ approximation to the revenue-optimal auction for a single buyer whose values for $m$ items are drawn from a distribution of support $\leq m$ over* matroid-based *valuations.*[4]

Importantly, note that a $1/\text{support}(D)$-approximation is trivial, so Theorem 13 rules out essentially any non-trivial approximation. To best understand our proof of Theorem 13, we must first overview the proof approach of [9] for their positive results. [9] establishes an approximation-preserving black-box reduction from revenue-maximization when all valuations come from class $\mathcal{V}$ (refer to this problem as MDMDP($\mathcal{V}$), formal definition in Section 2) to maximizing the difference of two functions in $\mathcal{V}$ (refer to this problem as ODP($\mathcal{V}$), formal definition in Section 2). For simple classes like additive valuations, ODP is easy, and this yields their positive results. For complex valuation classes like submodular, [9] establish that ODP is hard, and this intuition drives their inapproximability result.

However, [9] lacks a formal reduction from ODP to MDMDP. Using intuition that ODP(submodular) is hard, they directly construct hard MDMDP instances. Our next main result is a reduction from ODP to MDMDP. That is, the [9] reduction is essentially tight.

**Informal Theorem 2** (See Theorem 11). *For suitable $\mathcal{V}$, there is a black-box reduction from ODP($\mathcal{V}$) to MDMDP($\mathcal{V}$) which is almost approximation-preserving.*

Theorem 11 allows us to reason exclusively about hardness of ODP, and conclude hardness for MDMDP. Our final result provides a clean framework to quickly establish when ODP($\mathcal{V}$) is inapproximable, and proves that ODP(matroid-based) is indeed inapproximable (and also that matroid-based is suitable for the reduction in Theorem 11).

### 1.1 Context and Related Work

The context in which to view our work is the following: [9] establishes a reduction from MDMDP to ODP, and our Theorem 11 establishes a reduction from ODP to MDMDP, so their reduction is tight. Moreover, ODP is now a simple lens through which one can study (tightly!) the computational complexity of mechanism design — revenue-maximizing Bayesian mechanism design for valuations in class $\mathcal{V}$ is *exactly* as hard as maximizing the difference of two functions in $\mathcal{V}$ ([9] provides one direction of the reduction, and we provide the other).

[1] We will formally specify in Section 2 how valuation functions are presented as input, but quickly note that all results in this line of work (including ours) are compatible with any standard input format, such as value oracles, demand oracles, or an explicit poly-sized circuit which computes either query.

[2] A valuation is $c$-demand if $v(S) := \max_{T \subseteq S, |T| \leq c} \{\sum_{i \in T} v(\{i\})\}$. Additive valuations are $m$-demand, and Unit-demand valuations are 1-demand.

[3] A function $v(\cdot)$ is submodular if $v(S \cap T) + v(S \cup T) \leq v(S) + v(T)$ for all $S, T$.

[4] A matroid-based valuation satisfies $v(S) := \max_{T \subseteq S, T \in \mathcal{I}} \{\sum_{i \in T} v(\{i\})\}$, where $\mathcal{I}$ is a matroid.

Two previous works prove hardness of revenue maximization in this model. As already discussed, [9] uses intuition from hardness of ODP(submodular) to directly construct hard instances for MDMDP(submodular). They even provide a partial framework for proving similar results (which we overview in Section 2). Our results improve these by replacing submodular with matroid-based (a vastly more restrictive class — it is a proper subset of gross substitutes), and by providing a true reduction from ODP to MDMDP.

The other known hardness result is from [23], who establish that MDMDP(OXS)[5] is NP-hard to solve *exactly*. Their proof does follow a principled framework, and they establish roughly that whenever ODP($\mathcal{V}$) is hard to solve exactly, that MDMDP($\mathcal{V}$) is hard to solve exactly as well. Their framework, however, is very clearly limited to exact hardness.[6] Our results improve these by providing an approximation-preserving framework, which allows for hardness of approximation.

There is also a long series of related work in the *independent items* model [4, 10–15, 24, 28, 31, 33]. Here, the distribution $D$ is not given explicitly by listing its support, but some form of sample access (or other concise description) is given instead. The restriction is that $D$ satisfies "independent items". We refer the reader to [31] for the general definition, but the example to have in mind is an additive buyer, where independent items simply means that $v(\{i\})$ is drawn independently of $v(\{j\})$ for all $i \neq j$. Observe that inputting such a distribution explicitly in our model would require input of size $\exp(m)$. So the way to reconcile works such as [16–18] (which prove hardness of exact optimization for a single additive/unit-demand buyer) with folklore LPs (which provide poly-time algorithms for exact optimization for a single additive/unit-demand buyer) is that the hardness results rule out $\text{poly}(m)$-time solutions, whereas the LPs run in time $\exp(m)$ (which is the size of the support of the distribution). Similarly, the way to reconcile our results (which prove strong hardness of approximation for matroid-based) with [11, 15, 31] (which provides constant-factor approximations for a single subadditive buyer) is that these constant-factor approximations require the independent items assumption.

### 1.2 Summary and Roadmap

Our main results complete the picture for the [9] reduction, establishing an approximation-presering reduction from ODP to MDMDP, and moreover that MDMDP is inapproximable within non-trivial factors as soon as we move beyond $c$-demand valuations. Section 2 formally states the problems we study, and recaps prior work [9] in more detail. Section 3 establishes a clean framework to prove when ODP is inapproximable. Section 4 provides our reduction and concludes with our main theorem statements. The appendix contains all omitted proofs, along with some examples demonstrating interesting facts about some of our tools along the way.

## 2 PRELIMINARIES

### 2.1 Valuation Functions and Input

A *valuation function* takes as input a set $S$ of items and outputs a value $v(S)$. Throughout the paper, $[m]$ will denote the "base set" of $m$ items (although our reductions will create additional items). For ease of notation, and to help the reader parse parameters, we will w.l.o.g. scale all valuation functions so that $v(S) \in \mathbb{N}$ for all $S$ (this is w.l.o.g. when $v(S)$ is rational for all $S$, which is itself w.l.o.g. due to $\varepsilon$-truthful-to-truthful reductions of, e.g., [31, Theorem 5.2]). Moreover, all valuations considered in this paper are normalized ($v(\varnothing) = 0$), monotone ($v(S \cup T) \geq v(S)$ for all $S, T$), and

[5] OXS is defined in Section 2.

[6] This is because their produced MDMDP instances always have support 2, and it is trivial to get a 1/2-approximation on instances of support 2.

have no trivial items ($v(S) > 0$ for all $S \neq \emptyset$).[7] Finally, when $S$ is a random variable, we will abuse notation and let $v(S) := \mathbb{E}_S[v(S)]$. Below are the two main classes of valuations that our main results reference. Appendix E contains definitions of related classes of interest. Note that OXS $\subsetneq$ Matroid-based $\subsetneq$ Gross Substitutes $\subsetneq$ Submodular.

- OXS: there is a weighted bipartite graph $G$ with nodes $L = [m]$ on the left and $R$ on the right. $v(S)$ is the size of the max weight matching using nodes $S$ on the left and $R$ on the right. When all weights are 1 or 0, call this binary OXS.
- Matroid-based: let $\mathcal{I}$ be independent sets of a matroid on $[m]$, and $w_i$ be weights for each $i \in [m]$. Then $v(S) := \max_{T \subseteq S, T \in \mathcal{I}} \{\sum_{i \in T} w_i\}$. When each $w_i = 1$, call this matroid-rank.

*Representing Valuation Functions.* All problems we consider require a valuation function to be "input." Our theorem statements will be precise about what input models are assumed, although our results hold for most reasonable input models not explicitly discussed.

- Value oracle: each $v(\cdot)$ is given via a black box which takes as input a set $S$ and outputs $v(S)$.
- Demand oracle: each $v(\cdot)$ is given via a black box which takes as input a vector $\vec{p}$ of prices and outputs a set in $\arg\max_T \{v(T) - \sum_{i \in T} p_i\}$.
- Explicit input: each $v(\cdot)$ is given explicitly via a circuit or Turing machine which takes as input a set $S$ and outputs $v(S)$. We will only consider succinct representations (that is, circuits/Turing machines which are of polynomial size/runtime for inputs of size $m$).

### 2.2 Formal Problem Statements

The first problem we study is simply Bayesian revenue maximization, but we will be precise with how the input is specified to correctly place it with prior work. We use the language of [9] when possible to draw connections, although we will drop unnecessary parameters.

MDMDP($\mathcal{V}$) — Multi-Dimensional Mechanism Design Problem for class $\mathcal{V}$:
Input: an explicit distribution $D$ over $k$ valuation functions in $\mathcal{V}$ (given by listing all $v(\cdot)$ in the support, and the probability with which they are drawn from $D$).
Output: for each $v(\cdot)$ in the support of $D$, a (possibly randomized) set $S_v$ and a price $p_v$ such that $v(S_v) - p_v \geq v(S_w) - p_w$ for all $v, w$ in the support of $D$.
Objective: maximize $\sum_v \Pr[v \leftarrow D] \cdot p_v$, the expected revenue.
Approximation: a solution guarantees an $\alpha$-approximation if $\sum_v \Pr[v \leftarrow D] \cdot p_v \geq \alpha \cdot \text{OPT}$.

The lens by which we study MDMDP is the following standard optimization problem.

ODP($\mathcal{V}$) — Optimize Difference Problem for class $\mathcal{V}$:
Input: two functions $v(\cdot)$ and $w(\cdot)$, both in $\mathcal{V}$.
Output: a (possibly randomized) set, $S$.
Objective: maximize $v(S) - w(S)$.

[7] We will confirm that no trivial items is w.l.o.g. once we define our formal problems.

Approximation: a solution guarantees an $\alpha$-approximation if $v(S) - w(S) \geq \alpha \cdot \text{OPT}$.[8][9]

The final problem we study was introduced by [9], and shown to have connections to MDMDP. SADP essentially provides a list of related ODPs, and allows a solution to any of them.

SADP($\mathcal{V}$) — Solve Any Differences Problem for class $\mathcal{V}$:
Input: a finite list of functions $v_1(\cdot), \ldots, v_k(\cdot)$, all in $\mathcal{V}$.
Output: a (possibly randomized) set, $S$.
Objective: for some $j \in [k-1]$, have $S$ maximize $v_j(S) - v_{j+1}(S)$.
Approximation: a solution guarantees an $\alpha$-approximation if there exists a $j \in [k-1]$ such that $v_j(S) - v_{j+1}(S) \geq \alpha \cdot \max_T \{v_j(T) - v_{j+1}(T)\}$.

### 2.3 Recap of [9]

Finally, we briefly recap the main tools/results from [9] which are relevant for this paper. Recall that the focus of this paper is on hardness, and we establish our results with just a single buyer. Therefore, we will not recap the results of [9] in their full multi-buyer generality, but just focus on the single-buyer implications. Below, for any valuation class $\mathcal{V}$, $\mathcal{V}^*$ denotes its *conic closure*. That is, $\mathcal{V}^*$ denotes the closure of $\mathcal{V}$ under non-negative linear combinations. Many natural valuation classes (e.g. submodular, XOS, subadditive, additive) are already closed under conic combinations, but others (unit-demand, $c$-demand, OXS, matroid-based, gross substitutes) are not.

**Theorem 1** ([9]). *For all $\mathcal{V}$, there is a poly-time, approximation-preserving black-box reduction from MDMDP($\mathcal{V}$) to ODP($\mathcal{V}^*$). That is, an $\alpha$-approximation algorithm for MDMDP($\mathcal{V}$) (in any input model) exists using* poly($\textit{support}(D), m$) *black-box queries to an $\alpha$-approximation algorithm for ODP($\mathcal{V}^*$) (in that same input model), and additional runtime* poly($\textit{support}(D), m$).

For a single buyer, the positive applications of Theorem 1 are not particularly impressive, and imply only that MDMDP can be solved exactly whenever buyers are $c$-demand (which could alternatively be deduced by a simple linear program) — their main positive results are for multiple buyers. Again, our main result is an approximation-preserving reduction in the other direction, from ODP to MDMDP.

As referenced above, [9] also proves hardness of approximation for MDMDP(submodular). We make use of their machinery, which requires several definitions to precisely state. Without repeating the entire [9] hardness of approximation, it is perhaps impossible to motivate why these *precise* definitions are relevant, as they are technical in nature. However, we do give intuition to parse what the definitions are stating, and roughly the role they serve in prior work.

The first definition, Compatibility, is given below. Immediately afterwards, we provide context to help parse the condition.

[8] Observe above that if both $v(\cdot)$ and $w(\cdot)$ are subadditive, and $v(T) = 0$, then $v(S \cup T) = v(S)$ for all $S$, while $w(S \cup T) \geq w(S)$ for all $S$. Therefore, it is without loss to remove the items in $T$ from consideration. Similarly, if $w(T) = 0$, then $w(S \cup T) = w(S)$ for all $S$, while $v(S \cup T) \geq v(S)$ for all $S$. Therefore, it is without loss to solve ODP after removing $T$, and then add all items in $T$ back at the end. We therefore will pre-process any input to ODP by first finding all items $i$ such that $v(\{i\}) = 0$ and removing them, and all items $i$ such that $w(\{i\}) = 0$ and removing them (to add back later). Therefore, it is indeed w.l.o.g. to assume no trivial sets.

[9] We will also assume w.l.o.g. that there exists an $S$ for which $v(S) > w(S)$. This is without loss because if we have an algorithm $A$ which succeeds only on such instances, we can first run this algorithm on an arbitrary instance to get a set $T$, and check if $v(T) \geq w(T)$. If so, then output this (and either we were in a case where the algorithm succeeds, or the optimum is 0 and the algorithm succeeds anyway). If not, then output $\varnothing$ (because $v(T) < w(T)$ would prove that our algorithm failed, and therefore we are in a case where the optimum must be 0). So we will also assume this w.l.o.g. for our future reductions.

**Definition 1** (Compatibility)**.** We say that a list of valuation functions $V = (v_1, \ldots, v_k)$ and a list of (possibly randomized) sets $X = (X_1, \ldots, X_k)$ are *compatible* if:

- $X$ and $V$ are cyclic monotone. That is, the welfare-maximizing matching of valuations $V$ to allocations $X$ (that is, which maximizes $\sum_{i=1}^{k} v_i(X_{M(i)})$) is to match $X_i$ to $v_i$ for all $i$.
- For any $i < j$, the welfare-maximizing matching of valuations $v_{i+1}, \ldots, v_j$ to allocations $X_i, \ldots, X_{j-1}$ is to match allocation $X_\ell$ to valuation $v_{\ell+1}$ for all $\ell$.

One should parse Compatibility as a stronger condition than cyclic monotonicity (indeed, the first bullet is precisely cyclic monotonicity). Cyclic monotonicity requires that a particular matching on the complete bipartite graph is max-weight. Compatibility requires that a particular matching on several proper subgraphs are also max-weight. Intuitively, the particular restrictions in bullet two assert that higher-indexed allocations are "better", and higher-index valuations are "more important" (and therefore, the welfare-maximizing allocation always gives "better" allocations to "more important" valuations).

The next technical definition is a restriction on potential inputs to SADP. Intuitively, this condition serves the following purpose: the [9] reduction from SADP to MDMDP will take an input to SADP, and pass it on to MDMDP, and one part of their proof needs to establish the existence of a high-revenue solution to the produced MDMDP instance (so that any $\alpha$-approximation must also produce high revenue). $C$-compatibility suffices for this (but we will not attempt to explain further why this is the case, and refer the reader to [9] for more detail).

**Definition 2** ($C$-compatible)**.** A list of valuation functions $(v_1, \ldots, v_k)$ is *$C$-compatible* if there exist integers $1 = Q_1 < \ldots < Q_k$, all at most $2^C$, and allocations $(X_1, \ldots, X_k)$ such that:

- For all $\ell \in [k-1]$, $X_\ell \in \arg\max\{v_\ell(S) - v_{\ell+1}(S)\}$.
- $(Q_1 \cdot v_1, \ldots, Q_k \cdot v_k)$ is compatible with $(X_1, \ldots, X_k)$.

The final technical definition in their reduction is balanced-ness. This condition is used in their reduction to guarantee quality of approximation for the original SADP instance. The intuition to have in mind is that when given as input to MDMDP a distribution of support $k$, it is trivial to get a $1/k$-approximation simply by targeting the valuation in the support which maximizes $v([m]) \cdot \Pr[v \leftarrow D]$ (and setting price $v([m])$ for $[m]$, and no other options). Put another way, it's possible to guarantee a $1/k$ (or comparably poor) approximation to MDMDP without engaging with the instance at all. The purpose of their $d$-balanced property guarantees that sufficiently good approximations to the MDMDP instance produced by their reduction must actually engage the initial SADP instance.

**Definition 3** ($d$-balanced)**.** A list of functions $(v_1, \ldots, v_k)$ is $d$-balanced if $v_k([m]) \leq d \cdot (v_\ell(X) - v_{\ell+1}(X))$ for all $\ell \in [k-1]$, $X = \arg\max_T\{v_\ell(T) - v_{\ell+1}(T)\}$.

Following the intuition of the preceding paragraph, $d$-balanced aims to upper bound the revenue a seller could get on a SADP instance by simply selling $[m]$ to $v_k(\cdot)$ and ignoring everything else. With these two definitions, we may state the reduction of [9] from SADP to MDMDP:[10]

[10]This is indeed a correct statement of [9] Theorem 7 after chasing through their SADP choice of parameters.

**Theorem 2** ([9]). *Let $A$ be an $\alpha$-approximation algorithm for MDMDP($\mathcal{V}$). Then a solution to any $C$-compatible instance $(v_1, \ldots, v_k)$ of SADP($\mathcal{V}$) can be found in polynomial time plus one black-box call to $A$. The solution has the following properties:*

- *(Quality) If the SADP input is $d$-balanced, then the solution is an $(\alpha - \frac{(1-\alpha)d}{k-1})$-approximation.*
- *(Complexity) If for all $i$, $v_i([m]) \leq 2^b$ then $w([m]) \leq 2^{b+C}$ for all $w(\cdot)$ input to $A$. Moreover, all probabilities input to $A$ can be written as the ratio of two integers at most $2^{2C}$.*

The intended application of Theorem 2 is then to find a hard instance of SADP which is (a) $(\ll k)$-balanced — this guarantees that the resulting guarantee on SADP is close to $\alpha$, and (b) poly$(m)$-compatible — this guarantees that the input passed on to MDMDP blows up by only a poly$(m)$ factor. The appealing feature of Theorem 2 is that it suffices to establish that MDMDP(submodular functions) is inapproximable within any polynomial factor, by directly constructing a hard instance of SADP with $k = \text{poly}(m)$ valuations which is $(\ll k)$-balanced and poly$(m)$-compatible.

The unappealing feature of Theorem 2 is that SADP is not a particularly natural problem to think about, nor are the compatibility/balanced properties (hence, the need for a substantial preliminary section just to state their result). Additionally, ODP is a very special case of SADP whose solution suffice for MDMDP but Theorem 2 only establishes that MDMDP is hard when SADP is hard for large values of $k$.

Our work addresses both shortcomings. Theorem 11 provides a formal statement, which essentially replaces SADP with ODP (truly establishing that the [9] reduction is tight), and removes any compatibility/balanced requirements on the input instance (allowing greater ease of application to valuation classes significantly more restrictive than submodular).

## 3 WHEN IS ODP HARD?

Before diving into the technical part of our reduction, we'd first like a clean way to reason about valuation classes for which ODP is hard. Becuase ODP has a mixed-sign objective, one naturally expects that it is either solvable exactly in poly-time, or hard to even distinguish whether the optimum is non-zero (although there are sometimes exceptions to this intuition [21]). What's not immediately clear is how rich $\mathcal{V}$ needs to be before ODP becomes unsolvable in poly-time (e.g. it is solvable for additive functions, but not for submodular, what about in between?). In this section, we show that even ODP(binary OXS) is inapproximable. We begin with a general construction of hard instances:

**Definition 4** (Perturbable). For $v, S$, define $v_S(T) := v(T)$ if $S \neq T$, and $v_S(S) := v(S) - 1$. Class $\mathcal{V}$ is $(x, y)$-*perturbable* if there exists a $v(\cdot) \in \mathcal{V}$ such that $v_S(\cdot) \in \mathcal{V}$ for $x$ distinct $S$, and $v([m]) = y$.

$\mathcal{V}$ is efficiently $(x, y)$-perturbable if further there is a computationally-efficient bijection from the $x$ sets to $[x]$, and there is a poly-sized circuit/poly-time Turing machine which computes $v(\cdot)$.

The parameter $y$ will not be used in this section, and is not necessary to establish when ODP is hard, but we will need to reference the parameter in the technical parts of our reduction. Lemma 3 establishes that ODP is hard for highly-perturbable ($(x, y)$-perturbable for large $x$) classes. Intuitively, this is because the $x$ ODP instances of the form $(v, v_S)$ each have disjoint solutions, but are hard to distinguish. Proofs of the following two lemmas appear in Appendix A.

**Lemma 3.** *Let $\mathcal{V}$ be $(x, y)$-perturbable. Then any value- or demand-oracle algorithm for ODP($\mathcal{V}$) which guarantees an $\alpha$-approximation* for any *$\alpha > 0$ with probability $q$ makes $\geq \frac{qx-1}{2}$ queries.*

*If $\mathcal{V}$ is efficiently $(x, y)$-perturbable for $x = exp(m)$, then no randomized, poly-time algorithm for explicit input guarantees an $\alpha$-approximation for any $\alpha > 0$ w.p. $1/\text{poly}(m)$, unless NP $\subseteq$ RP.*

**Lemma 4.** *Binary OXS is efficiently* $(\binom{m}{m/2}, m/2)$*-perturbable.*[11]

Lemmas 3 and 4 together immediately conclude that ODP(Binary OXS) is inapproximable within any non-zero factor with poly$(m)$ value or demand queries, or within poly-time unless NP $\subseteq$ RP.

## 4 FROM ODP TO SADP

Now that we know ODP is hard even for fairly basic combinatorial valuations, we wish to leverage this to establish that the particular SADP instances needed for Theorem 2 are also hard. Again, the goal of this section is to provide a reduction from ODP to SADP, while also chasing through the parameters it implies for a full reduction from ODP to MDMDP. Before providing our reduction, let's get some intuition for the main challenges through some simple failed attempts. Recall that an $\alpha$ approximation for MDMDP implies an $(\alpha - \frac{(1-\alpha)d}{k-1})$ approximation for SADP. We want this to be as close to $\alpha$ as possible, so the properties we care about are:

- How balanced is the resulting SADP instance? (Smaller $d$ is better).
- How many functions are input to the SADP instance? (Larger $k$ is better).
- Is the instance poly$(m)$-compatible? (Necessary for the reduction to be valid — otherwise it will produce inputs to MDMDP of super-polynomial size).

**Failed Attempt 1.** The obvious first attempt is to just observe that ODP is a special case of SADP and simply "reduce" from ODP to SADP by copying the original instance itself into Theorem 2. This reduction is clearly correct, but let's examine what parameters we get from this reduction, using our $(\binom{m}{m/2}, m/2)$-perturbable $v$ as a representative instance:

- In that construction, $w([m]) = m/2$, and $\max_T\{v(T) - w(T)\} = 1$, so it is $(m/2)$-balanced.
- There are two functions input to ODP, so $k = 2$.

From above, $d/(k-1) = m/2$. So even if the instance were poly$(m)$-compatible, an $\alpha$-approximation for MDMDP wouldn't even imply a non-trivial guarantee for any $\alpha \leq \frac{1}{1+2/m}$. Recall again the intuition for this: in the [9] reduction, $k$ will be the size of the support of the produced $d$ for MDMDP. If $k = 2$, it's trivial to get a $1/2$-approximation without engaging SADP at all. So in order to get stronger hardness of approximation, we need much larger $k$.

**Failed Attempt 2.** Here is a next attempt, which takes $k$ as large as desired. Let $v, w$ be the given ODP instance, and define $v_\ell(\cdot) := (k - \ell) \cdot v(\cdot) + (\ell - 1) \cdot w(\cdot)$. Observe that for any $\ell$, $v_\ell(\cdot) - v_{\ell+1}(\cdot) = v(\cdot) - w(\cdot)$, so *every* consecutive difference is solving exactly the ODP instance we care about, and an $\alpha$-approximation for this SADP$(v_1, \ldots, v_k)$ will indeed yield an $\alpha$-approximation for ODP$(v, w)$. Let's check the parameters when we apply Theorem 2 with our hard ODP instance:

- $w([m]) = m/2$, and our $v_k([m]) = (k-1) \cdot w([m]) = (k-1)m/2$. Also, $\max_T\{v(T) - w(T)\} = 1$, so $\max_T\{v_\ell(T) - v_{\ell+1}(T)\} = 1$ for all $\ell$, and the instance is $(k-1)m/2$-balanced.
- We can set $k$ as we please.

Again, even if the instance were poly$(m)$-compatible, an $\alpha$-approximation for MDMDP still wouldn't imply non-trivial guarantees for any $\alpha \leq \frac{1}{1+2/m}$ (because still $d/(k-1) = m/2$). What we learn from these first two failed attempts is that we need to somehow let $k$ be large in our construction (have many functions), while simultaneously not letting $v_k([m])$ grow too big.

**Failed Attempt 3.** Our final failed attempt will also demonstrate one of our key insights: introduce extra items in the SADP instance. Specifically (for this construction), if we are initially given an

[11] So every superclass of Binary OXS (including Matroid-based) is efficiently $(\binom{m}{m/2}, m/2)$-perturbable as well.

ODP instance on $m$ items, and wish to produce a SADP instance with $k$ valuation functions, add an additional $k$ items. Call the initial items $M$ and the additional items $K$. For any set $S \subseteq M \cup K$, write it as $S = S_M \sqcup S_K$ (where $S_M = S \cap M, S_K = S \cap K$).[12] Define $v_\ell(S)$ to be equal to $v(S_M)$ if and only if $|S_K| \geq \ell$, and $w(S_M)$ if and only if $|S_K| \leq \ell - 1$ (note that this is a somewhat bizarre valuation function, which essentially treats the items in $K$ as a bit to decide whether the items in $M$ are valued using $v(\cdot)$ or $w(\cdot)$). It is not hard to see that any $\alpha$-approximation on this SADP instance implies an $\alpha$-approximation on our desired ODP instance. Let's again compute the parameters of the reduction for our hard ODP instance.

- $w([m]) = m/2$, and our $v_k([m]) = w([m]) = m/2$. Also, it is not hard to see that for a given $\ell$, $\max_T\{v_\ell(T) - v_{\ell+1}(T)\} = 1$. So the instance is $(m/2)$-balanced.
- We can set $k$ as we please.

Now, we're in good shape with parameters: we could (for example) set $k = m^c$ for any constant $c$, and establish that any $m^{-c+1}$-approximation for MDMDP instances with $m^c$ items implies a $(> 0)$-approximation for ODP on $m$ items, which we ruled out in Section 3. The catch is that our produced SADP instance has *bizarre* valuation functions. So all we have done is shown that MDMDP(bizarre valuation class) is hard.

### 4.1 Our Reduction

The previous examples illustrate the reasoning one needs to go through to analyze a reduction, and also identify some of the technical challenges. The two main (good) ideas we saw from the failed attempts were (a) introducing extra items and (b) "truncating" the valuations in a way that let $k$ grow large while keeping $d$ small. The challenge we did not yet overcome was how to do this in a way that preserves membership in interesting valuation classes, and we did not attempt to address Compatibility. We'll present our construction in two steps. The first adds additional items, but does not yet truncate.

**Reduction Step One: Scaled Disjoint Unions.**
Input: $v(\cdot), w(\cdot)$, both in $\mathcal{V}$ (input to ODP($\mathcal{V}$)), $k$ (desired number of valuations for SADP).
Output: $v_1(\cdot), \ldots, v_k(\cdot)$. Each $v_\ell(\cdot)$ takes as input subsets of $(k-1)m$ items. Partition these items into $M_1 \sqcup \ldots \sqcup M_{k-1}$, with each $|M_\ell| = m$. For any $S \subseteq M_1 \sqcup \ldots \sqcup M_{k-1}$, write it as $S := S_1 \sqcup \ldots \sqcup S_{k-1}$ (with each $S_\ell := S \cap M_\ell$). Define $v_\ell(S) := \sum_{i \geq \ell}(k+i) \cdot v(S_i) + \sum_{i<\ell}(k+i) \cdot w(S_i)$.

That is, we make $k-1$ copies of the $m$ items, have each $v_\ell(\cdot)$ behave either as $v$ or $w$ (scaled) on each copy, and then sum them. Observe that $v_\ell(\cdot)$ switches from $v$ to $w$ exactly at copy $\ell$, and that each $v_\ell(\cdot)$ is a *disjoint union* of valuations in $\mathcal{V}$, formally defined below. This operation happens to preserve membership in the valuation classes we care about.

For scaling, observe that all classes we consider except for Binary OXS and matroid-rank (because they insist on binary values) are closed under non-negative scaling. The scaling in each partition will not become relevant until the very end when we need to prove that the produced instance is poly$(m)$-compatible.

[12] As $S_M \cap S_K = \emptyset$, $S_M \sqcup S_K = S_M \cup S_K$. We use the disjoint union notation here (and in several other places) to remind the reader that these sets are always disjoint.

**Definition 5** (Disjoint Union of Valuations)**.** The *disjoint union* of two valuations $v(\cdot)$ and $w(\cdot)$ each on items $M$ creates two copies $M_1 \sqcup M_2$, and outputs a valuation $z(\cdot)$ such that $z(S) := v(S \cap M_1) + w(S \cap M_2)$.

**Fact 1.** *Additive, Binary OXS, OXS, Matroid-rank, Matroid-Based, Gross Substitutes, Submodular, XOS, and Subadditive valuations are closed under Disjoint Union. Unit-demand/c-demand are not.*

So we're in good shape so far in the sense that if we start with (say) matroid-based valuations in our ODP instance, we get back matroid-based valuations in our SADP instance. But we have not yet made any progress towards bounding the balanced-ness. Our next step is to properly truncate the valuations to make the valuations more balanced. We define two truncations below, which may inspire different related work. However, only Item Truncation is necessary for our main result, so we will focus the remaining presentation and formal statements on Item Truncation.

**Reduction Step Two: Item Truncation.**
Input: $v_\ell(\cdot)$, a valuation function and $y$, a desired truncation.
Output: $v'_\ell(\cdot)$, defined so that $v'_\ell(S) := \max_{T \subseteq S, |T| \leq y}\{v_\ell(T)\}$.

**Alternate Reduction Step Two: Value Truncation.**
Input: $v_\ell(\cdot)$, a valuation function and $x$, a desired truncation.
Output: $v'_\ell(\cdot)$, defined so that $v'_\ell(S) := \min\{v_\ell(S), x\}$.

We prove in Appendix B that matroid-based valuations are closed under item truncation, as this is the class/truncation we'll use for our main results. We state afterwards which other classes are closed under each truncation (proofs omitted, as we will not use these results elsewhere).

**Lemma 5.** *The class of matroid-rank valuations is closed under item truncation for any $y$.*

**Fact 2.** *Unit-demand, Submodular, XOS, and Subadditive are closed under value truncation for all $x$. Additive, $k$-demand, Binary OXS, OXS, Matroid-rank, Matroid-based, Gross Substitutes are not (there exist $x$, and a function $v(\cdot)$ in each of these classes for which value-truncating $v(\cdot)$ at $x$ is no longer in that class).*

**Fact 3.** *Unit-demand, Matroid-rank, Gross Substitutes, XOS, and Subadditive are closed under item truncation for all $y$.*[13] *Additive, $k$-demand, Binary OXS, OXS, and Submodular are not.*

Again, recall that Lemma 5 is all that's necessary for our main results (as hardness for matroid-rank implies hardness for all superclasses). Facts 2 and 3 are included for the sake of better understanding the truncation operations. One curious conclusion of Fact 3 is that the property of being closed under item-truncation "skips" submodular functions. Note also that value-truncation preserves submodularity but not Gross Substiutes, whereas item-truncation preserves Gross Substitutes but not submodularity.

[13] Of these, the claim for Gross Substitutes (like most properties of Gross Substitutes) is non-trivial to establish, and uses the fact that Gross Substitutes functions are *well-layered* [27].

### 4.2 Correctness, Balancedness, and Compatibility

Now, we want to establish that our reduction correctly solves the initial ODP instance, and also examine the parameters of the produced SADP instance. Our full reduction takes an ODP instance and first puts it through the Scaled Disjoint Union reduction with parameter $k$ to get $k$ valuations each on $(k-1)m$ items. Then, it puts each of these valuations through item truncation with parameter $m$. For a given ODP instance $(v, w)$, call the resulting SADP instance $IT(v, w)$. Appendix B contains a proof of Lemma 6, and Appendix C contains analogous claims for value truncation.

**Lemma 6.** *Let $S$ be an $\alpha$-approximation to the SADP instance $IT(v, w)$. Then with an additional $O(k^2 m)$ runtime/value queries (to the item-truncated values), an $\alpha$-approximation to the ODP instance $(v, w)$ can be found.*

Next, we want to see how well-balanced the instance is. Importantly, observe that the bound in Lemma 7 does not grow with $k$.

**Lemma 7.** *For any ODP instance $(v, w)$, the instance $IT(v, w)$ is $2mw([m])$-balanced.*

So at this point we have a reduction from ODP to very balanced instances of SADP, which also preserve valuation class membership for matroid-based valuation functions. Our last steps are to ensure that we can indeed compute value queries for our generated functions, and also to ensure that the resulting instance is poly$(m)$-compatible. Lemma 8 below is necessary for the following reason: Lemma 6 requires value queries *on values generated for $IT(v, w)$*. However, our input to ODP only provides access to $v, w$, which may not suffice for this.

**Definition 6.** Say that a list of valuation functions $V$ is *query-sufficient* for a list $W$ if (a) it is possible to execute value queries for every function in $W$ using polynomially-many value queries for functions in $V$ and (b) it is possible to write succinct circuits/Turing machines to compute value queries for every function in $W$ using succinct circuits/Turing machines computing value queries for every function in $V$.

Matroid-based $v, w$, due to connections with greedy algorithms, are indeed query-sufficient.

**Lemma 8.** *When $v, w$ are matroid-based, $v, w$ are query-sufficient for $IT(v, w)$.*

Lemma 8 confirms that if our ODP instance is given via explicit input or value queries, then this is enough to simulate value queries on each $v_\ell(\cdot)$ we construct. The last step is compatibility. We'll begin with a general sufficient condition for an instance to be Compatible, and finally establish that our instances satisfy this condition.

**Lemma 9.** *Let $v_1, \ldots, v_k$ be such that there exists allocations $X_1, \ldots, X_k$ satisfying:*

*(1) For all $\ell$, $X_\ell \in \arg\max_S \{v_\ell(S) - v_{\ell+1}(S)\}$.*
*(2) $v_\ell(X_i) \leq C_1$ for all $\ell, i$.*
*(3) $v_\ell(X_\ell) \geq v_\ell(X_{\ell-1}) + 1 \geq v_\ell(X_i) + 2$, for all $\ell > 1, i \leq \ell - 2$.*

*Then $(v_1, \ldots, v_k)$ is $k \log_2(C_1)$-Compatible.*

Our last step is to establish that our $IT(v, w)$ instance satisfies the hypotheses of Lemma 9.

**Corollary 10.** *$IT(v, w)$ is $k \log_2(2k \max\{v([m]), w([m])\})$-Compatible.*

### 4.3 Putting Everything Together

With all the pieces now in place, we can now prove Theorem 11. We briefly note that Theorem 11 also holds after replacing item truncation with value truncation (and the proof simply replaces Lemma 6 with Lemma 14).

**Theorem 11.** *Let $\mathcal{V}$ be closed under non-negative scaling, disjoint union, and item truncation, and let $(v, w)$ be query-sufficient for $IT(v, w)$ whenever $v, w \in \mathcal{V}$. Let also $A$ be an $\alpha$-approximation algorithm for MDMDP($\mathcal{V}$) using either value oracles or explicit input. Then for any integer $k \geq 2$, given black-box access $A$, there is a $\mathrm{poly}(km)$-time algorithm for ODP($\mathcal{V}$) making a single black-box call to $A$ plus an additional $\mathrm{poly}(km)$ value queries to the ODP input with the following properties:*

- *(Quality) On ODP input $(v, w)$, the algorithm produces an $(\alpha - \frac{(1-\alpha)2mw([m])}{k-1})$-approximation.*
- *(Complexity) Each value $v_\ell(\cdot)$ input to $A$ takes as input subsets of $[(k-1)m]$, and satisfies $v_\ell([(k-1)m]) \leq (2k \max\{v([m]), w([m])\})^k$. Moreover, all probabilities input to $A$ can be written as the ratio of two integers at most $(2k \max\{v([m]), w([m])\})^{2k}$.*

Proof. To complete the proof, we just need to carefully track all the pieces we put together. The fact that $\mathcal{V}$ is closed under non-negative scaling, disjoint union, and item truncation guarantees that $IT(v, w)$ will only contain valuations in $\mathcal{V}$ (they will also take as input subsets of $[(k-1)m]$). By Lemma 6, any $\alpha$-approximation to the SADP instance $IT(v, w)$ yields an $\alpha$-approximation to the ODP instance $(v, w)$ (in poly-time, because $(v, w)$ is query-sufficient for $IT(v, w)$). So our goal is to solve the SADP instance $IT(v, w)$, which is an instance of SADP($\mathcal{V}$).

Lemma 7 establishes that $IT(v, w)$ is $2mw([m])$-balanced, and Corollary 10 establishes that $IT(v, w)$ is $k \log_2(2k \max\{v([m]), w([m])\}$-Compatible. Therefore, we can solve the SADP instance $IT(v, w)$ using Theorem 2, and the approximation/complexity guarantees match up. □

We can now apply Theorem 11 to establish that MDMDP($\mathcal{V}$) is hard for perturbable $\mathcal{V}$.

**Theorem 12.** *Let $\mathcal{V}$ be closed under non-negative scaling, disjoint union, and item truncation, and let $(v, w)$ be query-sufficient for $IT(v, w)$ whenever $v, w \in \mathcal{V}$.*

- *For any integer $k \geq 2$, if $\mathcal{V}$ is $(x, y)$-perturbable, then no better than a $\frac{2my}{k-1+2my}$-approximation for MDMDP($\mathcal{V}$) can be guaranteed with probability $q$ on instances with $(k-1)m$ items with fewer than $\frac{qx-1}{2} - \mathrm{poly}(km)$ queries the value oracle model.*[14]
- *For any $k = \mathrm{poly}(m)$, if $\mathcal{V}$ is efficiently $(x, y)$-perturbable for $x = exp(m)$, then no better than a $\frac{2my}{k-1+2my}$-approximation for MDMDP($\mathcal{V}$) can be guaranteed with probability $1/\mathrm{poly}(m)$ on instances with $(k-1)m$ items in the explicit input model, unless NP $\subseteq$ RP.*

Theorem 12 implies that MDMDP(matroid-based) is inapproximable within non-trivial factors.

**Theorem 13.** *For all $\varepsilon > 0$, there does not exist an algorithm making $\mathrm{poly}(m)$ value queries in the value oracle model, $\mathrm{poly}(m)$ demand queries in the demand oracle model, or a poly-time algorithm in*

[14] Observe that because of the $-\mathrm{poly}(km)$, this result only has bite when $qx \gg \mathrm{poly}(km)$.

*the explicit input model (unless NP ⊆ RP) guaranteeing an $1/m^{1-\varepsilon}$-approximation with probability at least $1/\mathrm{poly}(m)$ for MDMDP(matroid-based) on instances with m items and $m^{1-2\varepsilon/9}$ values in the support of D.*

## REFERENCES


[1] Saeed Alaei. 2011. Bayesian Combinatorial Auctions: Expanding Single Buyer Mechanisms to Many Buyers. In *the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

[2] Saeed Alaei, Hu Fu, Nima Haghpanah, and Jason Hartline. 2013. The Simple Economics of Approximately Optimal Auctions. In *the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

[3] Saeed Alaei, Hu Fu, Nima Haghpanah, Jason Hartline, and Azarakhsh Malekian. 2012. Bayesian Optimal Auctions via Multi- to Single-agent Reduction. In *the 13th ACM Conference on Electronic Commerce (EC)*.

[4] Moshe Babaioff, Nicole Immorlica, Brendan Lucier, and S. Matthew Weinberg. 2014. A Simple and Approximately Optimal Mechanism for an Additive Buyer. In *the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

[5] Patrick Briest, Shuchi Chawla, Robert Kleinberg, and S. Matthew Weinberg. 2010. Pricing Randomized Allocations. In *the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[6] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. 2012. An Algorithmic Characterization of Multi-Dimensional Mechanisms. In *the 44th Annual ACM Symposium on Theory of Computing (STOC)*.

[7] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. 2012. Optimal Multi-Dimensional Mechanism Design: Reducing Revenue to Welfare Maximization. In *the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

[8] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. 2013. Reducing Revenue to Welfare Maximization: Approximation Algorithms and other Generalizations. In *the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[9] Yang Cai, Constantinos Daskalakis, and S. Matthew Weinberg. 2013. Understanding Incentives: Mechanism Design becomes Algorithm Design. In *the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.

[10] Yang Cai, Nikhil Devanur, and S. Matthew Weinberg. 2016. A Duality Based Unified Approach to Bayesian Mechanism Design. In *Proceedings of the 48th ACM Conference on Theory of Computation(STOC)*.

[11] Yang Cai and Mingfei Zhao. 2017. Simple mechanisms for subadditive buyers via duality. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 170–183. https://doi.org/10.1145/3055399.3055465

[12] Shuchi Chawla, Jason D. Hartline, and Robert D. Kleinberg. 2007. Algorithmic Pricing via Virtual Valuations. In *the 8th ACM Conference on Electronic Commerce (EC)*.

[13] Shuchi Chawla, Jason D. Hartline, David L. Malec, and Balasubramanian Sivan. 2010. Multi-Parameter Mechanism Design and Sequential Posted Pricing. In *the 42nd ACM Symposium on Theory of Computing (STOC)*.

[14] Shuchi Chawla, David L. Malec, and Balasubramanian Sivan. 2015. The power of randomness in Bayesian optimal mechanism design. *Games and Economic Behavior* 91 (2015), 297–317. https://doi.org/10.1016/j.geb.2012.08.010

[15] Shuchi Chawla and J. Benjamin Miller. 2016. Mechanism Design for Subadditive Agents via an Ex Ante Relaxation. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*. 579–596. https://doi.org/10.1145/2940716.2940756

[16] Xi Chen, Ilias Diakonikolas, Anthi Orfanou, Dimitris Paparas, Xiaorui Sun, and Mihalis Yannakakis. 2015. On the Complexity of Optimal Lottery Pricing and Randomized Mechanisms. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*. 1464–1479. https://doi.org/10.1109/FOCS.2015.93

[17] Xi Chen, Ilias Diakonikolas, Dimitris Paparas, Xiaorui Sun, and Mihalis Yannakakis. 2014. The Complexity of Optimal Multidimensional Pricing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 1319–1328. https://doi.org/10.1137/1.9781611973402.97

[18] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. 2014. The Complexity of Optimal Mechanism Design. In *the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

[19] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. 2015. Strong Duality for a Multiple-Good Monopolist. In *the 16th ACM Conference on Economics and Computation (EC)*.

[20] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. 2017. Strong Duality for a Multiple-Good Monopolist. *Econometrica* 85, 3 (2017), 735–767.

[21] Constantinos Daskalakis, Nikhil Devanur, and S. Matthew Weinberg. 2015. Revenue Maximization and Ex-Post Budget Constraints. In *the 16th ACM Conference on Economics and Computation (EC)*.

[22] Constantinos Daskalakis and S. Matthew Weinberg. 2015. Bayesian Truthful for Job Scheduling from Bi-Criterion Approximation Algorithms. In *the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.
[23] Shahar Dobzinski, Hu Fu, and Robert D. Kleinberg. 2011. Optimal Auctions with Correlated Bidders are Easy. In *the 43rd ACM Symposium on Theory of Computing (STOC)*.
[24] Sergiu Hart and Noam Nisan. 2012. Approximate Revenue Maximization with Multiple Items. In *the 13th ACM Conference on Electronic Commerce (EC)*.
[25] Sergiu Hart and Noam Nisan. 2013. The menu-size complexity of auctions. In *the 14th ACM Conference on Electronic Commerce (EC)*.
[26] Sergiu Hart and Philip J. Reny. 2015. Maximizing Revenue with Multiple Goods: Nonmonotonicity and Other Observations. *Theoretical Economics* 10, 3 (2015), 893–922.
[27] Renato Paes Leme. 2017. Gross substitutability: An algorithmic survey. *Games and Economic Behavior* 106 (2017), 294–316. https://doi.org/10.1016/j.geb.2017.10.016
[28] Xinye Li and Andrew Chi-Chih Yao. 2013. On Revenue Maximization for Selling Multiple Independently Distributed Items. *Proceedings of the National Academy of Sciences* 110, 28 (2013), 11232–11237.
[29] Roger B. Myerson. 1981. Optimal Auction Design. *Mathematics of Operations Research* 6, 1 (1981), 58–73.
[30] Gregory Pavlov. 2011. Optimal Mechanism for Selling Two Goods. *The B.E. Journal of Theoretical Economics* 11, 3 (2011).
[31] Aviad Rubinstein and S. Matthew Weinberg. 2015. Simple Mechanisms for a subadditive buyer and applications to revenue monotonicity. In *Proceedings of the 16th ACM Conference on Electronic Commerce*.
[32] John Thanassoulis. 2004. Haggling over substitutes. *Journal of Economic Theory* 117 (2004), 217–245.
[33] Andrew Chi-Chih Yao. 2015. An n-to-1 bidder reduction for multi-item auctions and its applications. In *the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

## A OMITTED PROOFS FROM SECTION 3

Proof of Lemma 3. Let $v$ witness that $\mathcal{V}$ is $(x, y)$-perturbable, and consider being given the ODP input $(v, v_S)$, for some $S$ such that $v_S \in \mathcal{V}$ chosen uniformly at random. Then $S$ is the unique optimum, and the only set which guarantees an $\alpha$-approximation for any $\alpha > 0$. First consider any (randomized) value-oracle algorithm which makes at most $s$ queries. Using the principle of deferred decisions, first select the (random) prices to query *conditioned on all previous responses being consistent with* $v(\cdot)$. Then as only one possible query ($S$) will be inconsistent with $v(\cdot)$, and $S$ is chosen uniformly at random, we have that except with probability $s/x$, all queries are consistent with $v(\cdot)$. When all queries are consistent with $v(\cdot)$, the algorithm must make some guess, which will be correct with probability at most $1/(x-s)$. So the total success probability is at most $s/x+1/(x-s)$. Setting $s = \frac{qx-1}{2}$ has this always at most $q$.

Next, consider (randomized) demand-oracle algorithms which make at most $s$ queries. Observe first that the *only* difference between a demand query for $v$ and a demand query for $v_S$ can come on a vector of prices $\vec{p}$ for which the demand query for $v$ on $\vec{p}$ outputs $S$ (this is because if any set $T \neq S$ is $v$'s favorite, it is certainly $v_S$'s favorite. But if $S$ is $v$'s favorite, it may not be $v_S$'s favorite). So again use the principle of deferred decisions, and select the (random) sets to query conditioned on all previous responses being consistent with $v(\cdot)$. Then for each query made, there is at most one $S$ for which $v_S$ might have output something different. Therefore, except with probability $s/x$, all queries are consistent with $v(\cdot)$. Again, the algorithm's total success probability is therefore at most $s/x + 1/(x - s)$.

To see the final claim in the explicit input model, we develop a reduction from unique-SAT (3-SAT where the promise is that there is exactly one or zero satisfying assignments). Given an instance $X$ of 3-SAT, consider a circuit/Turing machine which takes as input an assignment and outputs yes if that assignment is satisfying (so the circuit would be poly-sized, and the Turing machine would run in poly-time). Define now a circuit/Turing machine for a valuation fuction $v^X(\cdot)$ which takes as input a set $S$ and outputs $v(S)$ if $S$ is not one of the $x$ perturbing sets. Otherwise, first map $S$ to $[x]$,

and then take the corresponding assignment $\phi$. If $\phi$ satisfies $X$, output $v(S) - 1$. Otherwise, output $v(S)$. Note that this entire procedure is computationally-efficient/can be done with a poly-sized circuit because $\mathcal{V}$ is efficiently perturbable.

Now, assume for contradiction that there is a randomized algorithm for ODP($\mathcal{V}$) which guarantees a non-zero approximation with probably at least $1/\text{poly}(m)$. Given an instance $X$ of unique-SAT, run the algorithm on $v^X(\cdot)$, and let $S$ denote the output set. Observe that by the unique-SAT promise, $v^X(\cdot) \in \mathcal{V}$. Check if the assignment corresponding to $S$ satisfies $X$, and if so, output "yes" (otherwise, output "no"). This algorithm will clearly never mistakenly guess that $X$ is satisfiable. To see that it correctly says yes with probability at least $1/\text{poly}(m)$ (which, as usual, can be amplified to $2/3$ or $1 - 1/\text{poly}(m)$ if desired), observe that whenever $X$ is satisfiable uniquely by $\phi$, that the set $S$ corresponding to $\phi$ is the *only* set guaranteeing an $\alpha$-approximation for any $\alpha > 0$. Therefore, this set must be output with probability at least $1/\text{poly}(m)$ by our algorithm, implying that we will output $\phi$ (and correctly guess yes) with probability at least $1/\text{poly}(m)$. But no such algorithm for unique-SAT can exist unless NP $\subseteq$ RP. □

Proof of Lemma 4. Consider the complete bipartite graph with $m$ item nodes on the left and $m/2$ nodes on the right. Let $v(\cdot)$ denote the binary OXS valuation corresponding to this graph. Then $v(S) := \min\{|S|, m/2\}$, and $v(\cdot)$ is efficiently computable. Observe also that if we pick any set $S$ of size $m/2$ and remove all edges from each left-hand node in $S$ to (say) the top right-hand node (so we remove a total of $m/2$ edges), then we still have a binary OXS function. Moreover, we claim that this function is now $v_S$ for the removed $S$.

Indeed, observe that for any set $T$ of size $< m/2$, there is still a perfect matching from $T$ to the RHS, so we'll still have $v(T) = |T|$. For any set $T$ of size $\geq m/2$, perhaps $T = S$. If so, then clearly the max-weight matching has size $m/2 - 1$, as desired. Otherwise, $T$ contains an element not in $S$, which has an edge to the top node. There are at least $m/2 - 1$ left-hand nodes remaining, which have edges to every other right-hand node, and therefore form a matching of size $m/2 - 1$, for a total matching of $m/2$.

Therefore, $v(\cdot)$ can be perturbed at any set of size $m/2$. We can index these sets lexicographically to get an efficient mapping from these sets to $[\binom{m}{m/2}]$, as desired. □

## B OMITTED PROOFS FROM SECTION 4

Proof of Lemma 5. To see the claim, consider the independent sets $\mathcal{I}$ for the matroid defining $v(\cdot)$. It is well-known that updating $\mathcal{I}_y := \{S \in \mathcal{I}, |S| \leq y\}$ results in $\mathcal{I}_y$ being a matroid.[15] It is now easy to see that $v(\cdot)$ item-truncated at $y$ is exactly the matroid-based valuation corresponding to $\mathcal{I}_y$ (with the exact same weights $w_e$ for all items $e$), so it is matroid-based as well. □

Proof of Lemma 6. Throughout the proof, we'll let $M$ denote one copy of the original $m$ items, $M_i$ denote the $i^{th}$ copy, and use variables $S, T, \ldots$ to denote subsets of $\sqcup_i M_i$, and variables $A, B, \ldots$ to denote subsets of $M$.

[15]This is also easy to check: Clearly $\mathcal{I}_y$ is downwards closed. For any two independent sets $S, T$ in $\mathcal{I}_y$, with $|S| < |T|$, the fact that $\mathcal{I}$ satisfies the augmentation property immediately implies that there exists $i \in T$ such that $S \cup \{i\} \in \mathcal{I}$. As $|T| \leq y$, $S \cup \{i\} \in \mathcal{I}_y$ as well, so $\mathcal{I}_y$ has the augmentation property too.

Let $S = S_1 \sqcup \ldots \sqcup S_{k-1}$ denote the $\alpha$ approximation to the SADP instance $IT(v, w)$. First, we wish to refine $S$ so that $|S| \leq m$, but it is still an $\alpha$-approximation. To do this, observe that if $|S| \leq m$ then we are done. Otherwise, for each $\ell$ there is a subset $T \subseteq S$ with $|T| = m$ and $v_\ell(S) = v_\ell(T)$. To find this $T$, go through each item $i \in S$ one-by-one and check whether $v_\ell(S \setminus \{i\}) = v_\ell(S)$. If so, remove $i$ and continue (because there is still a set $T$ of size $m$ inside $S \setminus \{i\}$ with $v_\ell(T) = v_\ell(S)$). If not, we know that every such $T$ must contain $i$, so keep it. Stop when $|S| = m$. Observe that this procedure must terminate, because any $i \notin T$ will be removed when processed (and that it can be implemented with only value queries to $v_\ell(\cdot)$).

Do the above process for each $\ell$, to get $k$ candidate sets $S^\ell$, each of size at most $m$. Next, simply output $\arg\max_\ell \{v(S^\ell_\ell) - w(S^\ell_\ell)\}$ as the solution to the ODP instance $(v, w)$. We claim this must be an $\alpha$-approximation.

Consider first the solution $S$, and the index $\ell$ for which $v_\ell(S) - v_{\ell+1}(S) \geq \alpha \cdot \max_T \{v_\ell(T) - v_{\ell+1}(T)\}$. First, obesrve that indeed $\max_T \{v_\ell(T) - v_{\ell+1}(T)\} \geq (k + \ell) \max_A \{v(A) - w(A)\}$. This is because one candidate for $T$ is to set $S_\ell = \arg\max_A \{v(A) - w(A)\}$, and $S_i = \emptyset$ for all $i \neq \ell$ (which results in exactly this difference). So we may now conclude that:

$$v_\ell(S) - v_{\ell+1}(S) \geq \alpha(k + \ell) \cdot \max_A \{v(A) - w(A)\}.$$

Next, we claim that we must have $v_\ell(S^\ell) - v_{\ell+1}(S^\ell) \geq v_\ell(S) - v_{\ell+1}(S)$. This is simply because $S^\ell \subseteq S$, but $v_\ell(S^\ell) = v_\ell(S)$ by definition. Therefore, we get:

$$v_\ell(S^\ell) - v_{\ell+1}(S^\ell) \geq \alpha(k + \ell) \cdot \max_A \{v(A) - w(A)\}.$$

Finally, observe that $|S^\ell| \leq m$, so we know that:

$$\begin{aligned} v_\ell(S^\ell) - v_{\ell+1}(S^\ell) &= \sum_{i \geq \ell} (k + i) v(S^\ell_i) + \sum_{i < \ell} (k + i) w(S^\ell_i) - \sum_{i > \ell} (k + i) v(S^\ell_i) - \sum_{i \leq \ell} (k + i) w(S^\ell_i) \\ &= (k + \ell) \cdot (v(S^\ell_\ell) - w(S^\ell_\ell)) \end{aligned}$$

Therefore, we may conclude further that:

$$v(S^\ell_\ell) - w(S^\ell_\ell) \geq \alpha \cdot \max_A \{v(A) - w(A)\}.$$

The output of our algorithm is at least as good as $S^\ell_\ell$, because we take the maximum of this over all potential $\ell$, so we get an $\alpha$-approximation. □

Proof of Lemma 7. No matter how large $k$ is, any set with $m$ items can get value from at most $m$ copies of $M$. Within each copy, the maximum possible is $2kw([m])$, so the total value for any set is at msot $2kmw([m])$. Also, for all $\ell$, $\max_S \{v_\ell(S) - v_{\ell+1}(S)\} \geq (k + \ell)$. So the instance is $2mw([m])$-balanced. □

Proof of Lemma 8. The challenge is that, when $|S| \gg m$, it's not clear how to find the best $T \subseteq S$ of size $m$. Indeed, it is not necessarily possible in poly-time when $v, w$ are submodular, and therefore some assumption on valuation classes is necessary.

But when $v, w$ are matroid-based, recall that each $v_\ell(\cdot)$ is matroid-based as well. This suggests that a greedy algorithm can find the best $T \subseteq S$ of size $m$. Indeed, the following algorithm works:

(1) For each $e \in S$, find $v_\ell(e)$. This can be done with a single value query to either $v(\cdot)$ or $w(\cdot)$.
(2) Sort the elements in $S$ in decreasing order of $v_\ell(e)$.
(3) Initialize $T = \emptyset$.
(4) Process elements $e \in S$ one at a time.
   (a) If $|T| = m$, stop.
   (b) Else, compute $v_\ell(T \cup \{e\}) - v_\ell(T)$. This can be done with two queries to either $v(\cdot)$ or $w(\cdot)$ because $|T| < m$.
   (c) If $v_\ell(T \cup \{e\}) - v_\ell(T) > 0$, add $e$ to $T$. Otherwise, don't.
(5) Output $\sum_{i \geq \ell}(k+i) \cdot v(T_i) + \sum_{i<\ell}(k+i) \cdot w(T_i)$.

The algorithm above finds the max-weight independent subset of $S$, for the matroid defining the matroid-based valuation $v_\ell(\cdot)$, and the correct weights. Therefore, it correctly computes $v_\ell(S)$. □

Proof of Lemma 9. Consider the given allocations $(X_1, \ldots, X_k)$ and the multipliers $Q_i := (C_1 + 1)^{(i-1)}$. Observe first that all multipliers are integers at most $2^{k \log_2(C_1)}$, as desired. Also, we are already given that $X_\ell \in \arg\max\{v_\ell(S) - v_{\ell+1}(S)\}$ for all $\ell$. So we just need to establish that $V = (Q_1 \cdot v_1, \ldots, Q_k \cdot v_k)$ is compatible with $X = (X_1, \ldots, X_k)$.

First, let's establish that $V$ and $X$ are cyclic monotone. We proceed inductively from $\ell = k$ down to 1 and prove that $v_\ell$ must be matched to $X_\ell$. Assume for inductive hypothesis that the max-weight matching must match $v_i$ to $X_i$ for all $i > \ell$ and consider now $\ell$. Then consider the weight contributed by the edge between $v_\ell$ and $X_\ell$, versus $v_\ell$ and any $X_i$, $i < \ell$. By property 3, the former edge contributes at least $(C_1 + 1)^{\ell-1}$ more than any other potential edge. At the same time, by property 2, the maximum possible contribution from *all* edges $< \ell$ is at most $\sum_{i=1}^{\ell-1}(C_1+1)^{i-1}C_1 \leq C_1(C_1+1)^{\ell-1}/C_1$. Therefore, the maximum possible weight we can get from edges not adjacent to $v_\ell$ is less than what we lose by not matching $v_\ell$ to $X_\ell$, and the max-weight matching must match $v_\ell$ to $X_\ell$.

A nearly-identical argument establishes that the max-weight matching of valuations $v_{i+1}, \ldots, v_j$ to allocations $X_i, \ldots, X_{j-1}$ is to match $X_\ell$ to $v_{\ell+1}$ for all $\ell$. Again proceed inductively from $\ell = j$ down to $i+1$, and assume for inductive hypothesis that the max-weight matching must match $v_a$ to $X_{a-1}$ for all $a > \ell$. Consider again the weight contributed by the edge between $v_\ell$ and $X_{\ell-1}$ versus $v_\ell$ and any $X_i$, $i < \ell - 1$. By property 3, the former edge contributes at least $(C_1 + 1)^{\ell-1}$ more than any other potential edge. At the same time, by property 2, the maximum possible contribution from *all* edges $< \ell$ is at most $\sum_{i=1}^{\ell-1}(C_1+1)^{i-1}C_1 \leq C_1(C_1+1)^{\ell-1}/C_1$. Therefore, the maximum possible weight we can get from edges not adjacent to $v_\ell$ is less than what we lose by not matching $v_\ell$ to $X_{\ell-1}$, and the max-weight matching must match $v_\ell$ to $X_{\ell-1}$.

This completes the proof that $(v_1, \ldots, v_k)$ is $k \log_2(C_1)$-Compatible. □

Proof of Corollary 10. Consider the allocations $X_\ell$ defined so that $X_\ell \cap M_\ell \in \arg\max_A\{v(A) - w(A)\}$, and $X_\ell \cap M_i = \emptyset$ for all $i \neq \ell$. Then we clearly satisfy hypothesis 1 in Lemma 9. To see this, recall that for any set $S$, $v_\ell(S) - v_{\ell+1}(S) \leq v_\ell(S \cap M_\ell) - v_{\ell+1}(S \cap M_\ell)$, and $X_\ell$ is the optimal set contained in $M_\ell$.

Hypothesis 2 is also clearly satisfied for $C_1 = 2k \max\{v([m]), w([m])\}$. This follows because $v_\ell(M_i) \leq 2k \max\{v([m]), w([m])\}$ for all $\ell$.

Hypothesis 3 is satisfied for the following reasons. First, let $A^* := \arg\max_A\{v(A) - w(A)\}$. We know that $v(A^*) - w(A^*) \geq 0$. We also know that $v_\ell(X_\ell) = (k+\ell) \cdot v(A^*)$, and $v_\ell(X_i) = (k+i) \cdot w(A^*)$ for all $i < \ell$. As $v(A^*) \geq w(A^*)$, we Hypothesis 3 holds as well.

By Lemma 9, we may conclude that $IT(v, v_S)$ is $k \log_2(2k \max\{v([m]), w([m])\})$-Compatible. □

Proof of Theorem 12. To see the first claim, assume for contradiction that such an algorithm existed. Then with Theorem 11, we get an algorithm for ODP($\mathcal{V}$) which achieves a non-trivial approximation on all instances $(v, v_S)$ with probability $q$ with a total of $\frac{qx-1}{2} - \text{poly}(km) + \text{poly}(km)$ value queries (the extra queries come from Lemma 6). By Lemma 3, this is not possible. So the original MDMDP($\mathcal{V}$) algorithm must not exist.

To see the second claim, assume again for contradiction that such an algorithm existed. Then by Theorem 11, this would imply a poly-time algorithm with a non-trivial approximation on all instances $(v, v_S)$ with probability $1/\text{poly}(m)$ with total runtime $\text{poly}(km) = \text{poly}(m)$. By Lemma 3, this is impossible unless NP $\subseteq$ RP. □

Proof of Theorem 13. This is a corollary of Theorem 12 and Lemma 4. As Binary OXS is $(\exp(m), m/2)$-perturbable, so is matroid-based. Therefore, Theorem 12 implies that MDMDP(matroid-based) cannot be approximated better than $\frac{m^2}{k-1+m^2}$ with probability $1/\text{poly}(m)$ on instances with $(k-1)m$ items and distributions of support $k$ for any $k = \text{poly}(m)$. Letting $k = m^{2/\varepsilon}$, let $m' = km$ denote the number of items. Then there are $k \leq (m^{2/\varepsilon+1})^{1-\varepsilon/3} = (m')^{1-\varepsilon/3}$ valuations in the support, and the best achievable approximation guarantee is $m^{2-2/\varepsilon} \leq (m^{2/\varepsilon+1})^{-1+3\varepsilon/2}$.

So setting $\varepsilon' = 3\varepsilon/2$, we see that with $m'$ items and $(m')^{1-2\varepsilon'/9}$ valuations in the support of $D$, we can't get better than a $(m')^{1-\varepsilon'}$ approximation.

The only missing step is to connect everything to the demand oracle model. Observe that because all valuations involved are matroid-based (and therefore Gross Substitutes), that in fact a demand oracle can be implemented with $\text{poly}(m)$ value queries. So if we could solve MDMDP(matroid-based) using demand queries, we could use a value oracle for the $(v, w)$ input to ODP to get a value oracle for the MDMDP instance as in the reduction, which is also good enough for demand queries. □

## C CLAIMS FOR VALUE TRUNCATION

Below, define the SADP instance $VT(v, w)$ to first use the Disjoint Union reduction, followed by the value-truncation reduction with parameter $2kv([m])$.

**Lemma 14.** *Let $S$ be an $\alpha$-approximation to the SADP instance $VT(v, w)$. Then with an additional $O(k)$ runtime/value queries (to the value-truncated values), an $\alpha$-approximation to the ODP instance $(v, w)$ can be found.*

Proof. Throughout the proof, we'll let $M$ denote one copy of the original $m$ items, $M_i$ denote the $i^{th}$ copy, and use variables $S, T, \ldots$ to denote subsets of $\sqcup_i M_i$, and variables $A, B, \ldots$ to denote subsets of $M$.

Let $S = S_1 \sqcup \ldots \sqcup S_{k-1}$ denote the $\alpha$ approximation to the SADP instance $VT(v, w)$ (the resulting SADP instance from the value-truncation reduction). Simply output $\arg\max_\ell \{v(S_\ell) - w(S_\ell)\}$ (or $\varnothing$ if they all yield a negative answer) as the solution to the ODP instance $(v, w)$. We claim this must be an $\alpha$-approximation.

Consider first the solution $S$, and the index $\ell$ for which $v_\ell(S) - v_{\ell+1}(S) \geq \alpha \cdot \max_T\{v_\ell(T) - v_{\ell+1}(T)\}$. First, observe that indeed $\max_T\{v_\ell(T) - v_{\ell+1}(T)\} \geq (k+\ell)\max_A\{v(A) - w(A)\}$. This is because one candidate for $T$ is to set $S_\ell = \arg\max_A\{v(A) - w(A)\}$, and $S_i = \varnothing$ for all $i \neq \ell$ (which results in at least this difference, because our truncation exceeds $2kv([m])$). So we may now conclude that:

$$v_\ell(S) - v_{\ell+1}(S) \geq \alpha(k+\ell) \cdot \max_A\{v(A) - w(A)\}.$$

Next, there are a few cases to consider. First, perhaps $v_{\ell+1}(S) = 2kv([m])$. In this case, $v_\ell(S) \leq 2kv([m])$, so the LHS is 0, which guarantees an $\alpha$-approximation. Therefore, $\varnothing$ also guarantees an $\alpha$-approximation, so our algorithm succeeds. Next, perhaps $v_{\ell+1}(S) < 2kv([m])$. In this case, we get that:

$$\begin{aligned} v_\ell(S) - v_{\ell+1}(S) &= v_\ell(S) - \sum_{i>\ell}(k+i)v(S_i) - \sum_{i\leq\ell}(k+i)w(S_i) \\ &\leq \sum_{i\geq\ell}(k+i)v(S_i) + \sum_{i<\ell}(k+i)w(S_i) - \sum_{i>\ell}(k+i)v(S_i) - \sum_{i\leq\ell}(k+i)w(S_i) \\ &= (k+\ell)\cdot(v(S_\ell) - w(S_\ell)) \end{aligned}$$

Therefore, we may conclude further that:

$$v(S_\ell) - w(S_\ell) \geq \alpha \cdot \max_A\{v(A) - w(A)\}.$$

The output of our algorithm is at least as good as $S_\ell$, because we take the maximum of this over all potential $\ell$, so we get an $\alpha$-approximation. □

**Lemma 15.** *For any ODP instance $(v, w)$, the instance $VT(v, w)$ is $2v([m])$-balanced.*

Proof. Value-truncation explicitly caps $v_k(M_1 \sqcup \ldots \sqcup M_{k-1})$ at $2kv([m])$. Also, for all $\ell$, $\max_S\{v_\ell(S) - v_{\ell+1}(S)\} \geq (k+\ell)$. So the instance is $2v([m])$-balanced. □

**Lemma 16.** *All $v, w$ are query-sufficient for $VT(v, w)$.*

Proof. Simply observe that computing a value query for a function $v_\ell(S_1 \sqcup \ldots \sqcup S_{k-1})$ simply requires computing either $v(S_i)$ or $w(S_i)$ for all $i$, summing them, and taking a minimum with $x$, all of which can be done in time $O(k)$. □

## D EXAMPLE: SUBMODULAR IS NOT CLOSED UNDER ITEM TRUNCATION

Here we provide an example of a submodular function $v$ and size parameter $y$ such that $v' := \max_{T \subseteq S, |T| \leq y}\{v(T)\}$. This is sufficient to prove that submodular functions are not closed under item truncation, as our transformation of $v$ is a special instance of item truncation with $y = 2$.

Table 1. Valuation function $v$

| Sets of size 1 | Value | Sets of size 2 | Value | Sets of size 3 | Value | Sets of size 4 | Value |
|---|---|---|---|---|---|---|---|
| {a} | 5 | {c,d} | 10 | {a,b,c} | 9 | {a,b,c,d} | 10 |
| {b} | 5 | {a,d} | 9 | {a,b,d} | 10 | | |
| {c} | 5 | {b,d} | 9 | {a,c,d} | 10 | | |
| {d} | 5 | {a,c} | 9 | {b,c,d} | 10 | | |
| | | {b,c} | 9 | | | | |
| | | {a,b} | 9 | | | | |

We will use the following definition for submodularity: for any set of items $X$ and any items $y, z \notin X$, $v(X \cup y \cup z) - v(X \cup y) \leq v(X \cup z) - v(X)$. We will prove this by considering all possible sizes for $X$. If $X = \emptyset$: for all items $y$ and $z$, $v(y \cup z) - v(y) \leq 5 = v(z) - v(\emptyset)$. If $|X| = 1$: for all items $y$ and $z$, $v(X \cup y \cup z) - v(X \cup y) \leq 1 < 4 \leq v(X \cup z) - v(X)$. Now consider the case where $|X| = 2$. If $z \neq d$, $X \cup y$ cannot be the set $\{a, b, c\}$. In this case, $v(X \cup y) = v(X \cup z) = 10$, and therefore $v(X \cup y \cup z) - v(X \cup y) = 0 \leq v(X \cup z) - v(X)$. If $z = d$, then $X \neq \{c, d\}$, and therefore $v(X) = 9$. $X \cup y$ must be equal to $\{a, b, c\}$ and $X \cup z \neq \{a, b, c\}$, so $v(X \cup y) = 9$ and $v(X \cup z) = 10$. Putting it all together, $v(X \cup y \cup z) - v(X \cup y) = 10 - 9 = 1 = v(X \cup z) - v(X)$.

The function is also monotone. Note that the only sets $(X, Y)$ where $|X| < |Y|$ but $v(X) > v(Y)$ are $(\{c, d\}, \{a, b, c\})$. In this case $X \not\subseteq Y$, so it does not violate monotonicity. Thus, $v$ represents a monotone submodular function (so there's nothing "weird" about our counterexample).

Now let us define a function $v'$ as $\max_{T \subseteq S, |T| \leq 2}\{v(T)\}$. Note that $v$ and $v'$ will differ in evaluation only over $\{a, b, d\}$, where $v(a, b, d) = 10$ and $v'(a, b, d) = 9$. All other subsets of size greater than 2 contain the subset $\{c, d\}$ with value 10.

$v'$ is not submodular. Note that

$$v'(a, b, c, d) - v'(a, b, d) = 1 > 0 = v'(a, b, c) - v'(a, b)$$

If we let $X = \{a, b\}$, $x = c$ and $y = d$, we see that $v'(X \cup x \cup y) - v'(X \cup y) > v'(X \cup x) - v'(X)$, thus contradicting the submodularity definition.

## E VALUATION CLASSES

The following valuation classes are referenced throughout the text. They are listed below in strictly increasing order of generality.

- Additive: for all $S$, $v(S) := \sum_{i \in S} v(\{i\})$. Unit-demand: for all $S$, $v(S) := \max_{i \in S} v(\{i\})$.
- $c$-demand: for all $S$, $v(S) := \max_{T \subseteq S, |T| \leq c}\{\sum_{i \in T} v(\{i\})$.
- OXS: there is a weighted bipartite graph $G$ with nodes $L = [m]$ on the left and $R$ on the right. $v(S)$ is the size of the max weight matching using nodes $S$ on the left and $R$ on the right. When all weights are 1 or 0, call this binary OXS.

- Matroid-based: let $\mathcal{I}$ be independent sets of a matroid on $[m]$, and $w_i$ be weights for each $i \in [m]$. Then $v(S) := \max_{T \subseteq S, T \in \mathcal{I}} \{\sum_{i \in T} w_i\}$. When each $w_i = 1$, call this matroid-rank.
- Gross Substitutes: Let $\vec{q}, \vec{p}$ be price vectors with $q_i \geq p_i$ for all $i$. Let $P := \{i, q_i = p_i\}$. For all $X \in \arg\max_S \{v(S) - \sum_{i \in S} p_i\}$, there exists $Y \in \arg\max_S \{v(S) - \sum_{i \in S} q_i\}$ with $(P \cap X) \subseteq Y$.
- Submodular: for all $S, T$, $v(S) + v(T) \geq v(S \cap T) + v(S \cup T)$.
- XOS:[16] There exist additive functions $w_1, \ldots$ such that for all $S$, $v(S) := \max_j \{w_j(S)\}$.
- Subadditive: for all $S, T$, $v(S) + v(T) \geq v(S \cup T)$.

[16] Equivalently, Fractionally Subadditive.