Single-Parameter Combinatorial Auctions with Partially Public Valuations

Gagan Goel, Chinmay Karande, and Lei Wang

Georgia Institute of Technology, Atlanta. {gagang, ckarande, lwang}@cc.gatech.edu

Abstract. We consider the problem of designing truthful auctions, when the bidders' valuations have a public and a private component. In particular, we consider combinatorial auctions where the valuation of an agent *i* for a set *S* of items can be expressed as $v_i f(S)$, where v_i is a private single parameter of the agent, and the function *f* is publicly known. Our motivation behind studying this problem is two-fold: (a) Such valuation functions arise naturally in the case of ad-slots in broadcast media such as Television and Radio. For an ad shown in a set *S* of ad-slots, f(S) is, say, the number of *unique* viewers reached by the ad, and v_i is the valuation per-unique-viewer. (b) From a theoretical point of view, this factorization of the valuation function simplifies the bidding language, and renders the combinatorial auction more amenable to better approximation factors. We present a general technique, based on maximal-in-range mechanisms, that converts any α -approximate truthful mechanisms which run in polynomial time and quasi-polynomial time, respectively.

1 Introduction

A central problem in computational mechanism design is that of combinatorial auctions, in which an auctioneer wants to sell a heterogeneous set of items \mathcal{J} to interested agents. Each agent *i* has a valuation function $v_i(.)$ which describes her valuation $v_i(S)$ for every set $S \subseteq \mathcal{J}$ of items. In its most general form, the entire valuation function is assumed to be private information which may not be revealed truthfully by the agents. Maximizing the social welfare in a combinatorial auction with an incentive-compatible mechanism is an important open problem. However, recent results [5, 4] have established polynomial lower bounds on the approximation ratio of maximal-in-range mechanisms - which account for a majority of positive results in mechanism design - even when all the valuations are assumed to be submodular. On the other hand, in the non-game-theoretic case, if all the agents' valuations are public knowledge and hence truthfully known, then we can maximize the social welfare to much better factors [6, 7, 16], under varying degree of restrictions on the valuations. In this paper, we introduce a model that lies in between these two extremes.

We wish to explore the setting when some inherent property of the items induces a common and publicly known *partial* information about the valuation function of the buyers. For instance, in position auctions in sponsored search, the agents' valuation for a position consists of a private value-per-click as well as a public click-through rate, that is known to the auctioneer. Another situation where such private/public factorization of valuations arises is advertisements in broadcast media such as Television and Radio. Suppose we are selling TV ad-slots on a television network. There are m ad-slots and n advertisers interested in them. Let us define a function $f: 2^{[m]} \to \mathbb{Z}_+$, such that for any set S of ad-slots f(S) is the number of *unique* viewers who will see the ad if the ad is shown on each slot in S^1 . If an advertiser i is willing to pay v_i dollars per unique viewer reached by her ad, then her total valuation of the set S is $v_i f(S)$.

With this background, we define the following class of problems:

¹ For a single ad-slot j, the function $f(\{j\})$ is nothing but the television rating for that slot as computed by rating agencies such as Nielsen. In fact, their data collection through set-top boxes results in a TV slot-viewer bipartite graph on the sample population, from which f(S) can be estimated for any set Sof ad slots.

SINGLE-PARAMETER COMBINATORIAL AUCTIONS WITH PARTIALLY PUBLIC VALUA-TIONS: We are given a set \mathcal{J} of m items and a global *public* valuation function² $f : 2^{\mathcal{J}} \to \mathbb{R}$. The function f can either be specified explicitly or via an oracle which takes a set S as input and returns f(S). In addition, we have n agents each of whom has a *private* multiplier v_i such that the item set S provides $v_i f(S)$ amount of utility to agent i. The goal is to design a truthful mechanism which maximizes $\sum_i v_i f(S_i)$, where $S_1 \cdots S_n$ is a partition of \mathcal{J} .

One can think of this model as combinatorial auctions with simplified bidding language. The agents only need to specify one parameter v_i as their bid. Moreover, our problem has deeper theoretical connections to the area of single-parameter mechanism design in general. For single-parameter domains such as ours, it is known that monotone allocation rules characterize the set of all truthful mechanisms. An allocation rule or algorithm is said to be monotone if the allocation parameter of an agent $(f(S_i)$ in our case) is non-decreasing in his reported bid v_i . Unfortunately, often it is the case that good approximation algorithms known for a given class of valuation functions are not monotonic. It is an important and well-known open question in algorithmic mechanism design to resolve whether the design of monotone algorithms is fundamentally harder than the non-monotone ones. In other words, it is not known if, for single-parameter problems, we can always convert any α -approximation algorithm into a truthful mechanism with the same factor. We believe that our problem is a suitable candidate to attack this question as it gives a lot of flexibility in defining the complexity of function f. From this discussion, it follows that the only lower bound known for the approximation factor of a truthful mechanism in our setting is the hardness of approximation of the underlying optimization problem.

Our Results and techniques We give a general technique which accepts any (possibly nontruthful) α -approximation algorithm for our problem as a black-box and uses it to construct a truthful mechanism with an approximation factor of $\Omega\left(\frac{\alpha}{\log n}\right)$. We also give a truthful mechanism with factor $\Omega(\alpha)$ which runs in time $O\left(n^{\log \log n}\right)$. Both these results are corollaries obtained by setting parameters appropriately in Theorem 1 to achieve desired trade-off between the approximation factor and the running time. Our results can also be interpreted as converting non-monotone algorithms into monotone ones for the above model.

Our mechanisms are maximal-in-range, i.e., they fix a range \mathcal{R} of allocations and compute the allocation $\mathbf{S} \in \mathcal{R}$ that maximizes the social welfare. The technical core of our work lies in careful construction of this range.

While the black-box algorithm may be randomized, our mechanism does not introduce any further randomization. Depending upon whether the black-box algorithm is deterministic or randomized, our mechanism is deterministically truthful or universally truthful respectively (See Section 2 for definitions). The approximation factor of our mechanism is deterministic (or with high probability or in expectation) if the black-box algorithm also provides the approximation guarantees deterministically (or with high probability or in expectation).

Note that we don't need to worry about how the public valuation function f is specified. This is plausible since the function is accessed only from within the black-box algorithm. Hence, our mechanism can be applied to any model of specification - whether it is specified explicitly or through a value or demand oracle - using the corresponding approximation algorithm from that model.

Submodular valuations arise naturally in practice from economies of scale or the law of diminishing returns. Hence, we make a special note of our results when the public valuation is submodular. Using the algorithm of [16] as black-box, our results imply a $\Omega(1/\log n)$ and $\Omega(1)$ approximation factors in polynomial time and quasi-polynomial time, respectively. We would like to note that the standard greedy algorithm for submodular welfare maximization is not monotone

² We do not make any explicit assumptions such as non-negativity or free disposal about the function f. We provide a method to convert any non-truthful black-box algorithm into a truthful mechanism. This black-box algorithm may make some implicit assumptions about f.

(See Appendix A for a simple example) and hence, not truthful. Similarly, the optimal approximation algorithm of [16] is also not known to be non-monotone. The best known truthful mechanism for combinatorial auctions with entirely private submodular valuations [6] has $\Omega(1/\sqrt{m})$ approximation factor.

Future Directions As shown in [5, 4], it seems that designing a truthful mechanism with good approximation factor for maximizing social welfare is a difficult problem. In light of this, our work suggests an important research direction to pursue in combinatorial auctions- to divide the valuation function into a part which is common among all the agents and can be estimated by the auctioneer and a part which is unique and private to individual agents.

Also, it would be interesting to see if for submodular public functions (or even more specifically, for coverage functions), which have concrete motivation in TV ad auctions, one can design a constant factor polynomial time truthful mechanism.

Related Work When agents have a general multi-parameter valuation function, the best known truthful approximation of social welfare in the value oracle model is $\Omega(\sqrt{\log m}/m)$ [9]. Under subadditive valuation functions, [6] gave $\Omega(1/\sqrt{m})$ -approximate truthful mechanism. It is known that no maximal-in-range mechanism making polynomially many calls to the value oracle can have an approximation factor better than $\Omega(1/n^{1/6})$ [5] even for the case of submodular valuation functions. A similar $\Omega(1/\sqrt{m})$ hardness result for maximal-in-range algorithms based on NP $\not\subset$ P/poly appears in [4]. See [3] for a comprehensive survey of the results, and [15, 4] for other more recent work. Previous work on the single-parameter case of combinatorial auctions have primarily focused on the single-minded bidders. In this setting, any bidder i is only interested in single set S_i and has a valuation v_i for it. Lehmann et al. [11] gave a truthful mechanism which achieves an essentially best-possible approximation factor of $\Omega(1/\sqrt{m})$. For other results in single-minded combinatorial auction, see [13, 1]. When the desired set is publicly known and only the valuation is private, [2] gave a general technique which converts any α -approximation algorithm into a truthful mechanism with factor $\alpha/\log(v_{max})$. This result is very much in spirit to our work, however the model and the techniques used in the two papers are very different. Similarly, [10] present a general framework which uses a gap-verifying linear program as black-box to construct mechanisms that are truthful in expectation.

For the non-truthful optimization, we note that our problem is hard up to a constant factor (see [12]) even when all the agents have private value equal to 1 and with common valuation function being submodular. For designing monotone algorithms from non-monotone algorithms in the Bayesian setting, see [8]. We also note that TV ad auctions are in use by Google Inc. (see [14]), although currently they treat the valuations for a set of ad-slots as additive with budget constraints, which yields a multi-parameter auction.

Organization: Section 2 provides a brief introduction to mechanism design with a few concepts relevant to our work. Readers familiar with design of truthful mechanisms can skip to Section 3 in which we state some basic properties and assumptions about single parameter combinatorial auctions with partially public valuations. Section 4 introduces our vector-fitting technique and in Section 4.1, we conduct a warm-up exercise by analyzing a simple mechanism. Section 5 presents our main result, a vector-fitting mechanism formalized by Theorem 1.

2 Preliminaries

In this section, we will outline the basic concepts in mechanism design relevant to our paper.

2.1 Truthfulness and Mechanism Design

Mechanism design attempts to address the game-theoretic aspect of optimization problems. Let \mathcal{A} be the set of alternatives, and $u_i(a)$ be the valuation of agent *i* if alternative $a \in \mathcal{A}$ is picked. In

a pure optimization setting, all the functions u_i 's are assumed to be known to the auctioneer, and a typical goal is to pick an alternative $a \in \mathcal{A}$ that maximizes $\sum_i u_i(a)$. But from a game-theoretic perspective, the agents may have an incentive to lie about their valuation function u_i , if it leads to a better alternative for them. This kind of strategizing often results in arbitrary behaviour from the agents, leading to a loss in the social welfare. Mechanism design tackles this issue by designing algorithms such that truthfully reporting their true valuation function is the dominant strategy for each agent, *i.e.* given any strategies by all the other agents, reporting one's true function maximizes the utility gained by this agent.

There are three notions of truthfulness that may be applicable:

- 1. **Deterministic truthfulness**: The mechanism must be deterministic and an agent maximizes her utility by reporting her true valuation, for any valuations of all other agents.
- 2. Universal truthfulness: A universally truthful mechanism is a probability distribution over deterministically truthful mechanisms.
- 3. **Truthfulness in expectation**: A mechanism is truthful in expectation if an agent maximizes her *expected utility* by being truthful.

Every deterministically truthful mechanism is universally truthful and every universally truthful mechanism is truthful in expectation. Hence, deterministic truthfulness is the strictest notion of truthfulness. As noted earlier, our mechanism may be deterministically or universally truthful depending upon whether the black-box α -approximation algorithm is deterministic or randomized.

2.2 Vickrey-Clarke-Grove and Maximal-in-range Mechanisms

The Vickrey-Clarke-Grove (VCG) mechanism is a pivotal result in the field of mechanism design to maximize social welfare. It works as follows: let a^* and a^*_{-i} be the alternatives which maximizes $\sum_j v_j(a)$ and $\sum_{j \neq i} v_j(a)$ respectively. Now define payment p_i of agent *i* to be $\sum_{j \neq i} v_j(a^*_{-i}) - \sum_{j \neq i} v_j(a^*)$. It is now not difficult to see that with this payment function, it is in best interest of every agent to report their true valuations, irrespective of what others report.

As useful as the VCG mechanism is, it cannot be applied in many scenarios where the underlying problem is hard. Solving the optimization problem approximately doesn't preserve the truthfulness always. To overcome this, maximal-in-range variant of the VCG mechanism is a useful technique which optimizes over a smaller range of allocations. That is, the set of allocations that the mechanism may ever produce - the *range* - is chosen to be a small subset of the space of all allocations. The range is chosen to balance the following trade-off: A larger range can yield better approximation but require greater computational complexity. Note that such a range needs to be defined combinatorially *without* any knowledge of the agents' valuations.

For example, the $\Omega(1/\sqrt{m})$ -approximate truthful mechanism from [6] is a maximal-in-range mechanism.

3 Notations and Basic Properties

By boldface \mathbf{v} , we will denote a vector of private multipliers of the agents, where v_i is the multiplier of agent *i*. For a constant $\beta \ge 0$, let $\beta \mathbf{v} = (\beta v_1, \beta v_2, ..., \beta v_n)$. By boldface \mathbf{S} , we will denote the vector of allocations, where S_i is the set of items allocated to agent *i*. We will overload the function symbol *f* to express the social welfare as: $f(\mathbf{v}, \mathbf{S}) = \sum_i v_i f(S_i)$. An allocation \mathbf{S} is optimal for a multiplier vector \mathbf{v} if it maximizes $f(\mathbf{v}, \mathbf{S})$.

We begin by observing two simple properties of our problem and its solutions: symmetry and scale-freeness. Our problem and its solutions are symmetric, *i.e.*, invariant under relabeling of agents in the following sense: Let **v** be any multiplier vector, **S** be any allocation and π be any permutation of [n]. Let **u** and **T** be such that $u_i = v_{\pi(i)}$ and $T_i = S_{\pi(i)}$. Then clearly, $f(\mathbf{v}, \mathbf{S}) = f(\mathbf{u}, \mathbf{T})$. The problem and its solutions are also invariant under scaling, since we have $f(\beta \mathbf{v}, \mathbf{S}) = \beta \cdot f(\mathbf{v}, \mathbf{S})$.

The above properties lead us to:

Observation 1 Without loss of generality, every multiplier vector \mathbf{v} has non-increasing entries $v_1 \geq v_2 \geq ... \geq v_n$ such that $\sum_i v_i = 1$.

Given a multiplier vector \mathbf{v} , let $A(\mathbf{v})$ be the optimal allocation for \mathbf{v} and $OPT(\mathbf{v}) = f(\mathbf{v}, A(\mathbf{v}))$. Moreover, if $f(\mathbf{v}, \mathbf{S}) \geq \alpha \cdot OPT(\mathbf{v})$ for some $\alpha \leq 1$ then the allocation \mathbf{S} is said to be α -optimal or α -approximate for \mathbf{v} .

We note a simple property of $A(\mathbf{v})$: Let \mathbf{v} be a multiplier vector with $v_1 \ge v_2 \ge \dots \ge v_n$. Let \mathbf{S} be any allocation. If \mathbf{T} is a permutation of \mathbf{S} such that $f(T_1) \ge f(T_2) \ge \dots \ge f(T_n)$, then $f(\mathbf{v}, \mathbf{T}) \ge f(\mathbf{v}, \mathbf{S})$. In particular, if $\mathbf{S} = A(\mathbf{v})$ then $f(S_1) \ge f(S_2) \ge \dots \ge f(S_n)$.

Finally, we assume the existence of a poly-time black-box algorithm that computes an α approximate allocation $B(\mathbf{v})$ for the multiplier vector \mathbf{v} . We express the performance guarantees
of our truthful mechanisms in terms of α and other parameters of the problem. Although the
output allocation \mathbf{S} of such an algorithm may not obey $f(S_1) \ge f(S_2) \ge ... \ge f(S_n)$, it is easy to
construct a non-decreasing permutation of \mathbf{S} which only improves the objective function value, as
discussed above.

Observation 2 Without loss of generality, any allocation **S** output by the black-box algorithm obeys $f(S_1) \ge f(S_2) \ge ... \ge f(S_n)$.

Henceforth, we enforce assumptions from Observation 1 and 2.

Definition 1 (u dominates w). We say that a multiplier vector **u** dominates **w** if there exists an index *i* such that for k < i, $u_k \ge w_k$ and for $k \ge i$, $u_k \le w_k$.

Lemma 1. If **u** dominates **w**, then $f(\mathbf{u}, \mathbf{S}) \ge f(\mathbf{w}, \mathbf{S})$ for any allocation **S** satisfying $f(S_1) \ge f(S_2) \ge ... \ge f(S_n)$.

Proof. For k < i, let $x_k = u_k - w_k$. Similarly for $k \ge i$, $y_k = w_k - u_k$. Then

$$\sum_{k=1}^{i-1} x_k - \sum_{k=i}^n y_k = \sum_{k=1}^n u_k - \sum_{k=1}^n w_k = 0$$

which means $\sum_{k=1}^{i-1} x_k = \sum_{k=i}^n y_k$. Since $f(S_{k_1}) \ge f(S_{k_2})$ whenever $k_1 < i \le k_2$,

$$f(\mathbf{u}, \mathbf{S}) - f(\mathbf{w}, \mathbf{S}) = \sum_{k=1}^{i-1} x_k f(S_k) - \sum_{k=i}^n y_k f(S_k) \ge 0$$

Staircase Representation: Suppose we represent a multiplier vector \mathbf{v} as a histogram, which consists of *n* vertical bars corresponding to $v_1, ..., v_n$, in that order from left to right. Since multiplier vectors have non-increasing components, such a histogram looks like a staircase descending from left to right (Refer to Figure 1 for an example). We will refer to it as the *staircase representation* of \mathbf{v} and use it mainly as a visual tool.



Fig. 1. The staircase representation of $\mathbf{v} = (v_1, ..., v_n)$.

4 Vector-Fitting Mechanisms

Consider the following candidate approach to single parameter combinatorial auctions with partially public valuations: Fix a set \mathcal{U} of some multiplier vectors. Using the black-box algorithm, compute an α -approximate allocation $\mathbf{B}(\mathbf{v})$ for each vector $\mathbf{v} \in \mathcal{U}$ and populate the range $\mathcal{R} =$ $\{ B(\mathbf{v}) : \mathbf{v} \in \mathcal{U} \}$. Run the maximal-in-range mechanism which given a multiplier vector \mathbf{v} , chooses the allocation $\mathbf{S} \in \mathcal{R}$ that maximizes $f(\mathbf{v}, \mathbf{S})$.

Let's consider the merits and demerits of this mechanism. If the input multiplier vector happens to be in \mathcal{U} , then the mechanism will indeed return an output allocation that is at least α -approximate. But we have no guarantees otherwise. If \mathcal{U} consisted of all possible vectors, we would have an α -approximate truthful mechanism that could be computationally infeasible due to the size of \mathcal{U} . We handle this trade-off with *vector-fitting*. The intuition behind vector-fitting is as follows: If two multiplier vectors \mathbf{u} and \mathbf{v} are 'very similar' to each other, then $B(\mathbf{u})$ and $B(\mathbf{v})$ should be 'similar' as well. In particular, $B(\mathbf{u})$ should be a reasonably good allocation for \mathbf{v} and vice versa.

Our mechanism will be the same as the candidate mechanism outlined above, except that we will construct the set of vectors \mathcal{U} very carefully. For any input vector of multipliers \mathbf{v} , we will guarantee that a reasonably similar vector \mathbf{v}' can be found in \mathcal{U} , and hence and allocation \mathbf{S}' is in the range \mathcal{R} with provably large objective value $f(\mathbf{v}, \mathbf{S}')$.

4.1 A Simple $\frac{\alpha}{\ln n}$ -factor Mechanism

In this section, we will conduct a warm-up exercise by applying the vector-fitting method to construct a simple $\frac{\alpha}{\ln n}$ -factor truthful mechanism. Recall that the vector-fitting method as outlined in Section 4 starts with a set \mathcal{U} of multiplier vectors. Our set \mathcal{U} is defined as $\mathcal{U} = \{ \mathbf{u}^j : 1 \leq j \leq n \}$ where \mathbf{u}^j is defined as follows:

$$u_i^j = \frac{1}{j}$$
 for $1 \le i \le j$, zero elsewhere

As before, for each $\mathbf{v} \in \mathcal{U}$, we compute an α -approximate allocation $B(\mathbf{v})$ and populate the range \mathcal{R} with it.

Let **v** be the input multiplier vector. Let $r_j = \sum_{k=1}^j v_j$ for $1 \le j \le n$ be the prefix sums of **v**. We define prefix vectors \mathbf{d}^j of **v** as:

$$d_i^j = \frac{v_i}{r_j}$$
 for $1 \le i \le j$, zero elsewhere

It is easy to verify that \mathbf{d}^{j} is a valid multiplier vector *i.e.*, it has non-increasing components and unit l_{1} norm.



Fig. 2. Expressing $f(\mathbf{v}, \mathbf{S})$ as horizontal cuts of the staircase.

Let $\mathbf{S} = A(\mathbf{v})$ be the optimal allocation for \mathbf{v} and \mathbf{T} be the allocation returned by our mechanism. For notational convenience, define $v_{n+1} = 0$. We start with $OPT(\mathbf{v}) = f(\mathbf{v}, \mathbf{S})$ and look at how horizontal sections under the staircase of \mathbf{v} contribute to it. See figure 2.

$$OPT(\mathbf{v}) = \sum_{i}^{n} v_{i}f(S_{i})$$

$$= v_{n}\sum_{i=1}^{n} f(S_{i}) + (v_{n-1} - v_{n})\sum_{i=1}^{n-1} f(S_{i}) + \dots + (v_{1} - v_{2})f(S_{1}) \quad (1)$$

$$= \sum_{j=1}^{n} \left[j(v_{j} - v_{j+1}) \sum_{i=1}^{j} \frac{f(S_{i})}{j} \right]$$

$$= \sum_{j=1}^{n} \left[j(v_{j} - v_{j+1}) \cdot f(\mathbf{u}^{j}, \mathbf{S}) \right]$$

$$\leq \sum_{j=1}^{n} \left[j(v_{j} - v_{j+1}) \cdot OPT(\mathbf{u}^{j}) \right]$$

$$\leq \sum_{j=1}^{n} \left[\alpha^{-1}j(v_{j} - v_{j+1}) \cdot f(\mathbf{u}^{j}, \mathbf{B}(\mathbf{u}^{j})) \right]$$

$$\leq \sum_{j=1}^{n} \left[\alpha^{-1}j(v_{j} - v_{j+1}) \cdot f(\mathbf{d}^{j}, \mathbf{B}(\mathbf{u}^{j})) \right] \quad (2)$$

$$\leq \sum_{j=1}^{n} \left[\frac{j(v_j - v_{j+1})}{\alpha r_j} \cdot f(\mathbf{v}, \mathbf{B}(\mathbf{u}^j)) \right]$$
(3)

$$\leq \sum_{j=1}^{n} \left[\frac{j(v_j - v_{j+1})}{\alpha r_j} \cdot f(\mathbf{v}, \mathbf{T}) \right]$$
(4)

$$=\frac{f(\mathbf{v},\mathbf{T})}{\alpha}\left[1+\sum_{j=2}^{n}\frac{v_{j}(r_{j}-jv_{j})}{r_{j}r_{j-1}}\right]$$
(5)

Equation (1) decomposes $f(\mathbf{v}, \mathbf{S})$ as the horizontal cuts of the staircase of \mathbf{v} (See Figure 2). (2) follows from the previous step by applying a simple structural property formalized by Lemma 1 to \mathbf{d}^{j} and \mathbf{u}^{j} . Equation (3) follows from (6) below, which is a simple restatement.

$$\sum_{i=1}^{j} \frac{v_i}{r_j} \cdot f(S_i) \leq \frac{1}{r_j} \sum_{i=1}^{n} v_i f(S_i) = \frac{f(\mathbf{v}, \mathbf{S})}{r_j}$$
(6)

Equation (5) is derived from the previous expression by simply rearranging the terms. Since $v_j \leq r_j/j$ and $r_j - jv_j \leq r_{j-1}$, we conclude that

$$1 + \sum_{j=2}^{n} \frac{v_j(r_j - jv_j)}{r_j r_{j-1}} \leq \sum_{j=1}^{n} \frac{1}{j} \leq \ln n$$

and

$$f(\mathbf{v}, \mathbf{T}) \ge \alpha \cdot \operatorname{OPT}(\mathbf{v}) / \ln n$$

An example that achieves the bound: We can differentiate each term of the summation in equation (5) to compute the values of v_j for which the term is maximized, so as to make the bound as loose as possible. Surprisingly, a single multiplier vector maximizes all the terms simultaneously! This vector is defined as $v_j = (\sqrt{j} - \sqrt{j-1})/\sqrt{n}$. Some calculations prove that for this multiplier vector, the summation is indeed $\Omega(\ln n)$.

5 The Main Result

In this section, we will use vector-fitting to obtain a general technique to convert a non-truthful approximation algorithm for single parameter combinatorial auctions into a truthful mechanism. This technique yields a range of trade-offs between the approximation factor and the running time of the algorithm. We will prove the following theorem:

Theorem 1. There exists a truthful mechanism for maximizing welfare in a single parameter combinatorial auction with partially public valuations that runs in time $O((\log_a n)^{\log_b n} \cdot \operatorname{poly}(m, n))$ and produces an allocation with total welfare at least $\frac{3\alpha}{4ab} \cdot \operatorname{OPT}(\mathbf{v})$ - where α is the approximation factor of the black-box optimization algorithm and a, b > 1 are parameters of the mechanism.

Setting a = b = 2 we get: (Henceforth, all logarithms are to base 2)

Corollary 1. There exists a $\frac{3\alpha}{16}$ -factor truthful mechanism running time $O((n^{\log \log n} \cdot \operatorname{poly}(m, n)))$, i.e. quasi-polynomial time.

Similarly, setting a = 2 and $b = \log n$ we get:

Corollary 2. There exists a truthful mechanism with factor $\Omega\left(\frac{\alpha}{\log n}\right)$ and polynomial running time.

When the public valuation f is submodular, we have $\alpha = \left(1 - \frac{1}{e}\right)$ and the above corollaries yield factors $\Omega(1)$ and $\Omega\left(\frac{1}{\log n}\right)$ respectively.

5.1 Constructing the Range \mathcal{R}

Overview: Recall the staircase representation of a multiplier vector \mathbf{v} , such as in Figure 1. Depending upon the entries of \mathbf{v} , the steps of the staircase may have varying heights. We can construct a discretization of the space of all multiplier vectors by restricting the values the height of any step can take. That is, we populate the initial set \mathcal{U} with all vectors whose components take values of the form b^{-k} for some constant b > 1 and for all $k \ge 0$. Now given any input vector \mathbf{v} , we can find a vector $\mathbf{u} \in \mathcal{U}$ such that u_i is at most a multiplicative factor b away from v_i . Thus, \mathbf{u} can serve as a vector 'similar' to \mathbf{v} . We need more complex machinery to ensure that the size of \mathcal{U} does not blow up, and that the vectors in \mathcal{U} still have unit norm.

Let a, b > 1 be suitably chosen parameters of the mechanism. Let $Q = \{ b^{-k} : 0 \le k < \log_b n \}$ be a set of values discretizing the interval $(\frac{1}{n}, 1]$ and q be the minimum element of Q. For a multiplier $v_i \ge q$, we define $\lfloor v_i \rfloor$ to be the largest element of Q that is no greater than v_i . For a multiplier vector \mathbf{v} we define the floor of $\mathbf{v}, \lfloor \mathbf{v} \rfloor$ as follows:

Definition 2 (Floor $\lfloor \mathbf{v} \rfloor$). The floor $\lfloor \mathbf{v} \rfloor$ of a multiplier vector \mathbf{v} is the vector \mathbf{u} constructed by Algorithm 1.

Algorithm 1: ConstructFloor

for i = 1 to n do $r \leftarrow \frac{\left(1 - \sum_{k=1}^{i-1} u_k\right)}{(n-i+1)};$ /* r is the minimum permissible value of u_i due to monotonicity. */ if $v_i \ge q$ and $\lfloor v_i \rfloor > r$ then $\mid u_i \leftarrow \lfloor v_i \rfloor;$ else $\mid or j = i \text{ to } n \text{ do}$ $\mid u_j \leftarrow r;$ break In short, to find the 'floor' of a multiplier vector, we successively round down the 'large' components into elements of Q, until we need to set all the remaining components equal due the monotonicity and unit norm requirement or only 'small' components are remaining. When represented as a staircase (Refer Figure 3), all the steps of $\lfloor \mathbf{v} \rfloor$ except the last one must have height that belongs to Q.



Fig. 3. Vertical fitting of v.

Observation 3 The floor of a vector \mathbf{v} is a valid multiplier vector itself, i.e. it has non-increasing components and unit l_1 norm. Moreover, \mathbf{v} dominates $\lfloor \mathbf{v} \rfloor$.

Proof. Refer to Appendix B.

Intuitively, the floor of a vector is (in a sense formalized by Lemma 2) 'similar' to the vector, and the similarity is parametrized by b.

Lemma 2. For any multiplier vector \mathbf{v} and allocation \mathbf{S} , $f(\lfloor \mathbf{v} \rfloor, \mathbf{S}) \geq \frac{3}{4b} \cdot f(\mathbf{v}, \mathbf{S})$.

Proof. Refer to Appendix C.

We will construct our preliminary set of vectors \mathcal{U}' as

 $\mathcal{U}' \;=\; \{ \; \mathbf{u} \;:\; \mathbf{u} = \lfloor \mathbf{v} \rfloor \; \text{for some multiplier vector } \mathbf{v} \; \}$

It turns out that \mathcal{U}' is too large for our purposes. Hence we construct a subset $\mathcal{U} \subseteq \mathcal{U}'$, which is small enough. Referring back to the staircase representation of a multiplier vector (Figure 3), we constructed \mathcal{U}' by discretizing the 'height' of each step - by fitting the vectors vertically. Since rounding down the components of \mathbf{v} might lead to many components of $\mathbf{u} = \lfloor \mathbf{v} \rfloor$ having the same value, \mathbf{u} also looks like a staircase, perhaps with 'wider' steps. Each step of \mathbf{u} may have any integral width - at most n.

We construct \mathcal{U} from \mathcal{U}' by further restricting how wide a step can be - by horizontal fitting (See Figure 4). We allow each step (except the last) to be of width $\lceil a^k \rceil$ for some integer $k \ge 0$ - where a > 1 is a suitably chosen parameter of the mechanism. To this end, we need to slightly formalize the staircase representation of a multiplier vector, which till now we only used as a visual aid. By a *step* of the staircase of \mathbf{v} , we will mean a maximal interval $[i_1, ..., i_2] \subseteq [1, ..., n]$ such that $v_{i_1} = v_{i_2}$. All the indices $i_1 \le i \le i_2$ will be said to belong to the step, whereas i_1 and i_2 and the first and last indices of the step. The *height* of the step is given by v_{i_1} and the *width* by $i_2 - i_1 + 1$.

Remark: Notice that just as a multiplier vector can be specified by the *n*-tuple $(v_1, ..., v_n)$, it can also be identified by specifying the height and width of each step of its staircase representation. In fact, specifying all but the last step of a staircase fixes the last step due to the unit norm requirement.

For a multiplier vector \mathbf{v} , we define the core $\overleftarrow{\mathbf{v}}$ of \mathbf{v} as:

Definition 3 (Core $\overleftarrow{\mathbf{v}}$). The core $\overleftarrow{\mathbf{v}}$ of a multiplier vector \mathbf{v} is the vector \mathbf{u} constructed by Algorithm 2.

Algorithm 2: ConstructCore

Operation of Algorithm 2: Each iteration of the while loop processes one step of \mathbf{v} and \mathbf{u} . i_1 and j_1 hold the first index of the current step of \mathbf{v} and \mathbf{u} respectively. r is the minimum height of the current step of \mathbf{u} by monotonicity. If $r \geq v_{i_1}$, then the requirement for unit l_1 norm forces us to introduce the last step of the staircase of \mathbf{u} . Otherwise, $[i_1, ..., i_2]$ is the current step of \mathbf{v} and we set the width of the current step of \mathbf{u} to be $\lceil a^k \rceil$.



Fig. 4. Horizontal fitting of v.

Observation 4 The core of a vector \mathbf{v} is a multiplier vector itself, i.e. it has non-increasing components and unit l_1 norm. Moreover, \mathbf{v} dominates $\overleftarrow{\mathbf{v}}$.

Proof. Refer to Appendix D.

Lemma 3. For any multiplier vector \mathbf{v} and allocation \mathbf{S} , $f(\mathbf{\nabla}, \mathbf{S}) \geq f(\mathbf{v}, \mathbf{S})/a$.

Proof. Refer to Appendix E.

We now define our set of vectors \mathcal{U} as follows: $\mathcal{U} = \{ \overleftarrow{\mathbf{v}} : \mathbf{v} \in \mathcal{U}' \}$. We populate the range \mathcal{R} of allocations as $\mathcal{R} = \{ B(\mathbf{v}) : \mathbf{v} \in \mathcal{U} \}$ where $B(\mathbf{v})$ is the α -approximate allocation returned by the black box algorithm.

5.2 Proof of Theorem 1

We run the following maximal-in-range mechanism: Given an input multiplier vector \mathbf{v} we return the allocation $\mathbf{T} \in \mathcal{R}$ that maximizes $f(\mathbf{v}, \mathbf{T})$. We need to prove that $f(\mathbf{v}, \mathbf{T}) \geq \frac{3\alpha}{4ab} \cdot \operatorname{OPT}(\mathbf{v})$

Let $\mathbf{S} = \mathbf{A}(\mathbf{v})$ be the optimal allocation for \mathbf{v} and $[\mathbf{v}]$ be the core of the floor of \mathbf{v} . Combining Lemmas 2 and 3, we conclude that $f([\mathbf{v}], \mathbf{S}) \geq \frac{3}{4ab} \cdot \operatorname{OPT}(\mathbf{v})$. Since $[\mathbf{v}] \in \mathcal{U}$, there exists an allocation $\mathbf{X} \in \mathcal{R}$ such that

$$f(\overleftarrow{[\mathbf{v}]}, \mathbf{X}) \geq \alpha \cdot \operatorname{OPT}(\overleftarrow{[\mathbf{v}]}) \geq \alpha \cdot f(\overleftarrow{[\mathbf{v}]}, \mathbf{S}) \geq \frac{3\alpha}{4ab} \cdot \operatorname{OPT}(\mathbf{v})$$
(7)

Since v dominates $\lfloor v \rfloor$ which in turn dominates $\lfloor v \rfloor$ (Refer to Observation 3 and 4), application of Lemma 1 yields:

$$f(\mathbf{v}, \mathbf{X}) \geq f(\lfloor \mathbf{v} \rfloor, \mathbf{X}) \geq f(\lfloor \mathbf{v} \rfloor, \mathbf{X})$$
 (8)

Using equations (7) and (8),

$$f(\mathbf{v}, \mathbf{T}) \geq f(\mathbf{v}, \mathbf{X}) \geq f(\overleftarrow{|\mathbf{v}|}, \mathbf{X}) \geq \frac{3\alpha}{4ab} \cdot \operatorname{OPT}(\mathbf{v})$$

The running time of the mechanism is established by Lemma 4, which finishes the proof of Theorem 1.

Lemma 4. $|\mathcal{R}| = O\left((\log_a n)^{\log_b n}\right)$

Proof. $|\mathcal{R}|$ is bounded by $|\mathcal{U}|$. \mathcal{U} consists of only those vectors which are cores of floors of some multiplier vectors. We have seen that each step of the staircase of $\mathbf{v} \in \mathcal{U}$ except the last must be of width $w = \lceil a^k \rceil$ for some integer k. Moreover, there can be only $|Q| = O(\log_b n)$ such steps and at most one of each height. We have also remarked that specifying all but the last step of a staircase fixes it. Therefore there can be at most $O((\log_a n)^{\log_b n})$ distinct staircases in \mathcal{U} .

References

- Aaron Archer, Christos Papadimitriou, Kunal Talwar, and Éva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pages 205–214, 2003.
- [2] Moshe Babaioff, Ron Lavi, and Elan Pavlov. Single-value combinatorial auctions and algorithmic implementation in undominated strategies. J. ACM, 56(1):1–32, 2009.
- [3] Liad Blumrosen and Noam Nisan. Algorithmic Game Theory, chapter 11. Cambridge University Press, 2007.
- [4] Dave Buchfuhrer, Shaddin Dughmi, Hu Fu, Robert Kleinberg, Elchanan Mossel, Christos Papadimitriou, Michael Schapira, Yaron Singer, and Chris Umans. Inapproximability for vcg-based combinatorial auctions. In SODA '10: Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete algorithms, pages 518–536, 2010.
- [5] Shahar Dobzinski and Noam Nisan. Limitations of vcg-based mechanisms. In STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, pages 338–344, 2007.
- [6] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, pages 610–618, 2005.
- [7] Shahar Dobzinski and Michael Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pages 1064–1073, 2006.
- [8] Jason D. Hartline and Brendan Lucier. Bayesian algorithmic mechanism design. In STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing, pages 301–310, 2010.
- [9] Ron Holzman, Noa Kfir-dahav, Dov Monderer, and Moshe Tennenholtz. Bundling equilibrium in combinatorial auctions. *Games and Economic Behavior*, 47(1):104–123, April 2001.

- [10] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pages 595–604, 2005.
- [11] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. J. ACM, 49(5):577–602, 2002.
- [12] Vahab Mirrokni, Michael Schapira, and Jan Vondrak. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In EC '08: Proceedings of the 9th ACM conference on Electronic commerce, pages 70–77, 2008.
- [13] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. *Games and Economic Behavior*, 64(2):612–631, 2008.
- [14] Noam Nisan, Jason Bayer, Deepak Chandra, Tal Franji, Robert Gardner, Yossi Matias, Neil Rhodes, Misha Seltzer, Danny Tom, Hal Varian, and Dan Zigmond. Google's auction for tv ads. In ICALP '09: Proceedings of the 36th International Collogquium on Automata, Languages and Programming, pages 309–327, 2009.
- [15] Christos Papadimitriou, Michael Schapira, and Yaron Singer. On the hardness of being truthful. In FOCS '08: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science, pages 250–259, 2008.
- [16] Jan Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing, pages 67–74, 2008.

A The Greedy Algorithm is not Monotone

The greedy algorithm in our setting works as follows: In each step it assigns one unallocated item j to a buyer i, where the pair (i, j) is chosen so as maximize the marginal gain in the objective function. That is, if buyer i had been allocated the set S of items before the current step, then $v_i (f(S \cup \{j\}) - f(S))$ is maximized.

We will construct an example that adheres to our formulation of the TV Ad auctions problem. Consider an instance with two advertisers i_1 and i_2 and three ad-slots j_1, j_2, j_3 . Suppose there are 10 viewers $k_1, ..., k_{10}$. Viewers k_1 to k_5 watch slot j_1, k_6 to k_{10} watch slot j_2 and k_3 to k_8 watch slot j_3 . The public function f in this case is the coverage function: for a set S of slots f(S) is the number of unique viewers who watch any slot in S. To prove that the greedy algorithm does not make monotonic allocations in this example, consider two cases:

- 1. $v_{i_1} = 1$ and $v_{i_2} = 1 + \epsilon$: In the first step, the greedy algorithm assigns the largest slot j_3 (with six viewers) to i_2 . In the next two steps, it assigns both j_1 and j_2 to i_1 . Therefore, the i_1 receives the set $\{j_3\}$ of total allocation value (not counting the private multiplier) 6.
- 2. $v_{i_1} = 1$ and $v_{i_2} = 1 \epsilon$: In the first step, the greedy algorithm assigns the largest slot j_3 (with six viewers) to i_1 . In the next two steps, it assigns both j_1 and j_2 to i_2 . Therefore, the i_1 receives the set $\{j_1, j_2\}$ of total allocation value (not counting the private multiplier) 10.

Clearly, i_2 receives a larger allocation at a lower private valuation. Therefore, the greedy algorithm is not monotone.

B Proof of Observation 3

The procedure to compute $\mathbf{u} = \lfloor \mathbf{v} \rfloor$ easily ensures the unit l_1 norm. Now to prove monotonicity by contradiction, assume that there exists *i* such that $u_i < u_{i+1}$. Since \mathbf{v} satisfies monotonicity, this can only happen if v_i was strictly rounded down to u_i and v_{i+1} was not. Therefore

$$1 - \sum_{k=1}^{i-1} u_k = \sum_{k=i}^n u_k = u_i + \sum_{k=i}^n u_k = u_i + (n-i)u_{i+1} > (n-i+1) \cdot u_i$$

This implies

$$\lfloor v_i \rfloor = u_i < \frac{1 - \sum_{k=1}^{i-1} u_k}{n - i + 1}$$

which is impossible since the right-hand side of the above inequality is the minimum value u_i could have been assigned.

To see that v dominates $\mathbf{u} = |\mathbf{v}|$, observe that if the index i does not belong to the last step of **u**, then v_i must have been rounded down to u_i , and therefore, $u_i \leq v_i$. Now consider the smallest i such that $u_i > v_i$. Then i must belong to the last step of **u**, and hence $u_i = u_i > v_i \ge v_i$ for any $j \geq i$.

\mathbf{C} Proof of Lemma 2

Define $\mathbf{u} = \lfloor \mathbf{v} \rfloor$. Let p be the highest index such that v_p is rounded down by the procedure that constructs \mathbf{u} , *i.e.* $u_p = \lfloor v_p \rfloor$ and $u_p > r = u_{p+1}$. Since, $\sum_{i=1}^p u_i \leq \sum_{i=1}^p v_i$, it is clear that p < n. Now for $i \leq p$, we have $u_i = \lfloor v_i \rfloor \geq v_i/b$. Consider two cases about v_{p+1} :

Case 1 - $v_{p+1} \ge q$: In this case, $u_{p+1} = r \ge \lfloor v_{p+1} \rfloor \ge v_{p+1}/b$. For $i \ge p+1$, we have $v_i \leq v_{p+1}$ and $u_i = u_{p+1}$ implying $u_i \geq v_i/b$. Therefore,

$$f(\mathbf{u}, \mathbf{S}) = \sum_{i=1}^{n} u_i f(S_i) \geq \frac{1}{b} \sum_{i=1}^{n} v_i f(S_i) = \frac{1}{b} \cdot f(\mathbf{v}, \mathbf{S})$$

Case 2 - $v_{p+1} < q$: Let $h = \sum_{i=1}^{p} v_i$ and $H = \left(\sum_{i=1}^{p} v_i f(S_i)\right) / f(\mathbf{v}, \mathbf{S})$. From the monotonicity

of **S**, we conclude that

$$H \cdot f(\mathbf{v}, \mathbf{S}) = \sum_{i=1}^{p} v_i f(S_i) \ge h \cdot f(\mathbf{v}, \mathbf{S})$$

and hence H > h.

Since $u_i \leq v_i$ for all $i \leq p$, and both **u** and **v** must have unit l_1 norm, we have $\sum_{i>p} u_i \geq v_i$ $\sum_{i>p} v_i = (1-h)$. Hence, $u_i \ge \frac{1-h}{n}$ for i > p. By definition, $v_i < q \le \frac{b}{n}$ for i > p. Together, these imply $u_i \ge (1-h)v_i/b$. Finally, using $H \ge h$, we conclude

$$\sum_{i>p} u_i f(S_i) \geq \frac{1-h}{b} \left(\sum_{i>p} v_i f(S_i) \right) \geq \frac{1-H}{b} \left[(1-H) f(\mathbf{v}, \mathbf{S}) \right]$$

Combining these pieces together, we get:

$$f(\lfloor \mathbf{v} \rfloor, \mathbf{S}) = \sum_{i=1}^{p} u_i f(S_i) + \sum_{i>p} u_i f(S_i)$$
$$\geq \frac{1}{b} \sum_{i=1}^{p} v_i f(S_i) + \frac{(1-H)^2}{b} \cdot f(\mathbf{v}, \mathbf{S})$$
$$= \frac{H + (1-H)^2}{b} \cdot f(\mathbf{v}, \mathbf{S}) \geq \frac{3}{4b} \cdot f(\mathbf{v}, \mathbf{S})$$

Proof of Observation 4 D

The algorithm to construct $\mathbf{u} = \mathbf{\overline{v}}$ itself easily ensures the unit norm. To prove monotonicity by contradiction, assume that there exists i such that $u_i < u_{i+1}$. This can only happen is i+1 is the first index of the last step of **u** and *i* is the last index of the penultimate step. Let *j* be the first index of the penultimate step. Then

$$1 - \sum_{k=1}^{j-1} u_k = \sum_{k=j}^n u_k = (i-j+1)u_i + (n-i)u_{i+1} > (n-j+1) \cdot u_i$$

This means

$$u_j = u_i < \left(1 - \sum_{k=1}^{j-1} u_k\right) / (n-j+1)$$

which is impossible since the right-hand side of the above inequality is the minimum value u_j could have been assigned.

To see that **v** dominates $\mathbf{u} = \overleftarrow{\mathbf{v}}$, observe that if the index *i* does not belong to the last step of **u**, then $u_i = v_j$ for some $j \ge i$, and hence $u_i = v_j \le v_i$. Now consider the smallest *i* such that $u_i > v_i$. Then *i* must belong to the last step of **u**. Therefore, $u_j = u_i > v_i \ge v_j$ for all $j \ge i$. \Box

E Proof of Lemma 3

Suppose the staircase of **v** has s_1 steps and that of $\mathbf{u} = \mathbf{\nabla}$ has s_2 steps. Then the following four properties follow directly from the algorithm:

1. $s_2 \leq s_1$

- 2. For $1 \le i < s_2$, the *i*'th step of **v** is at most *a* times as wide as the *i*'th step of **u** and both have the same height.
- 3. For $1 \le i \le s_2$, let i_1 and j_1 be the first indices of the *i*'th steps of **v** and **u** respectively. Then $i_1 \ge j_1$.
- 4. If [j, ..., n] is the last step of **u** then $u_i \ge v_i$ for $i \ge j$.

To prove the lemma, we will compare the the contributions of corresponding steps of the staircases of \mathbf{v} and \mathbf{u} to the objective functions.

For $i < s_2$, let $[i_1, ..., i_2]$ be the *i*'th step of \mathbf{v} , $[j_1, ..., j_2]$ be the *i*'th step of \mathbf{u} and $h = v_{i_1} = u_{j_1}$ be their common height. We have

$$\sum_{k=j_1}^{j_2} u_k f(S_k) = h \sum_{k=j_1}^{j_2} f(S_k) \ge h \sum_{k=i_1}^{i_i+j_2-j_1} f(S_k)$$

by the third property. The monotonicity of \mathbf{S} and the second property then imply

$$\sum_{k=j_1}^{j_2} u_k f(S_k) \geq \frac{1}{a} \sum_{k=i_1}^{i_2} v_k f(S_k)$$

So the *i*'th step of **v** contributes at most *a* times value to $f(\mathbf{v}, \mathbf{S})$ as the *i*'th step of **u** contributes to $f(\mathbf{u}, \mathbf{S})$, where $i < s_2$.

Finally by the fourth property, the step s_2 of **u** contributes more to $f(\mathbf{u}, \mathbf{S})$ than the corresponding contribution of steps $s_2, ..., s_1$ of **v** to $f(\mathbf{v}, \mathbf{S})$ combined. The result therefore follows. \Box