Brown University

Master's Thesis

Hypergraph Valuations with Restricted Overlapping

Gianluca Pane

supervised by
Dr. Amy Greenwald

May 15, 2017

Contents

Abstract

In a single-item auction, each bidder submits a valuation for the good and the auctioneer typically awards the item, or *good*, to the highest bidder, thereby maximizing the social welfare. If multiple goods are auctioned simultaneously, not only does the task of maximizing social welfare (winner determination) become more complex, but eliciting bids becomes complex as well. Since the goods can exhibit substitute or complementary relationships, a valuation over the set G of goods may require exponential storage (at worst, a value for each subset of G). This problem motivates the study of *bidding languages*: fully or near-fully expressive languages for expressing valuations which aim to be concise for some useful valuation classes. One such bidding language uses hypergraphs to represent valuations; each good is represented by a node and each *hyperedge* (incident to an arbitrary number of vertices) represents the marginal benefit or loss from obtaining a particular bundle of items. This representation is fully expressive, so in order to solve the winner determination problem, we will need to restrict the graph in some way. While Abraham et al (2012) and Feige et al (2014) restrict the *rank* of the hypergraph—the maximal degree of any hyperedge—we instead restrict the degree to which hyperedges are allowed to overlap. We obtain polynomial-time winner determination algorithms for small amounts of overlapping.

1 Introduction

In single-item auctions, the task of finding the social welfare-maximizing allocation is simple: just give the good to the highest bidder. In such environments, the challenge is to construct an *incentive compatible* mechanism—that is, a mechanism in which each bidder is incentivized to bid her true valuation. This way, the auctioneer can be confident that the highest bidder indeed values the item the most, thus achieving the maximum possible social welfare.

In combinatorial auctions, a set G of goods is auctioned off and a bidder can receive any subset of G. In this environment, maximizing social welfare is considerably more difficult. If the goods in G are independent—if for any bidder i and subset $S \subseteq G$, $v_i(S) = \sum_{j \in S} v_i(j)$—then we could just greedily allocate each item to the bidder who values it the most. However, there may be dependencies between the items. A kayak and a paddle, for example, may exhibit a *complementary* relationship, in which $v_i(S) > \sum_{j \in S} v_i(j)$. If instead S is a set of various models of kayaks, then it might be that $v_i(S) < \sum_{j \in S} v_i(j)$, in which case we say that the elements of S are *substitutes*.

In order to encompass all possible relationships between goods in G, it seems that for each bidder we must list a numerical value for her value of each subset of G. This is the most basic example of what we call a *bidding language*—a method of representing a valuation over a set of goods. We call it a *bidding* language because it is what each bidder will use to report their valuations to the auctioneer. All bidding languages must strike some balance between the following:

(1) Expressivity: How large is the space of valuation functions that are possible to represent in the language?

(2) Conciseness: How much space do the representations use?

(3) Interpretability: How long does it take to find the value of a subset? (This is the value query problem.)

(4) Winner determination: Can the winner determination problem be solved in polynomial time for any valuation instance in the language?

The proposed subset-value bidding language performs very well on the first quality, but quite poorly on the others. It is *fully expressive*—i.e. it can be used to represent *any* possible valuation over the goods—but it requires exponential space and would take exponential time to solve for a value query.

Since it is not possible to be fully expressive and, for instance, to solve the winner determination problem in polynomial time, another way of judging a bidding language is on its extensibility. Is it easy to impose restrictions on some fully expressive language and if so, can we solve value query or winner determination on the restricted languages?

In this paper, we use weighted *hypergraphs* to represent valuation functions. Each item is represented by a node and each *hyperedge* connected to some subset S of goods; vertex weights represent item values while hyperedge weights represent the marginal gain (or loss) from obtaining the corresponding subset in full. While this representation is fully expressive, it is considerably easier to impose useful restrictions compared to the subset-value bidding language.

Abraham, et al[1] and Feige, et al[7] both restrict the *rank* of the hypergraph, defined as the greatest number of items that can be associated with a hyperedge. When the rank is restricted to two and the graph is planar, Abraham et. al find a polynomial-time $(1+\epsilon)$-approximation of the optimal social welfare. More generally, they find a polynomial-time r-approximation for rank-r hypergraphs.

Abraham, et al also prove that their r-approximation is tight, so we will need a different method of restricting the hypergraphs to find useful results. Our approach in this paper is to allow hyperedges of arbitrary size, but restrict the degree to which the hyperedges can overlap with each other. This notion of "overlapping degree" is quite vague and lends itself to two different possible interpretations. One notion of overlapping is how "dense" a region can be; that is, for any item, how many different hyperedges all contain that one item. Another notion is how "connected" the hypergraph is; that is, for any hyperedge h, how many hyperedges overlap with h (i.e. have shared items). As it turns out, even if we restrict goods to lie in no more than two hyperedges, the winner determination problem is NP-hard, as we show in section 5. However, if we instead restrict each hyperedge to connect to no more than two other hyperedges, we obtain a polynomial-time algorithm for the optimal social welfare (section 6).[1]

2 Related Work

In this section, we review some of the major bidding languages introduced in the last twenty years. The aim of a bidding language is to achieve full or near-full expressivity but also to be concise

[1] In fact, the restriction we make is slightly more lenient, since we allow for arbitrary containment of hyperedges.

for typical or useful classes of valuations. Since it is not possible to simultaneously achieve this level of expressivity *and* solve the winner determination problem (or even the value query problem) in polynomial time, we will present restrictions of these languages under which polynomial-time winner determination algorithms exist.

2.1 Bidding Languages

One major class of these languages stem from boolean operators on *atomic bids*. We first introduce the natural bidding languages formed from ORs, XORs, and ORs-of-XORs of atomic bids[11], then briefly introduce two more complicated languages, $\mathcal{L}_{GB}$ and $TBBL$, which use a number of additional operators to achieve better conciseness. Then we will introduce the hypergraph and Maximum Over Positive Hypergraph representations of valuation functions.

Simple Boolean Operator-Based Languages

Before introducing boolean operator-based languages, we need to define the atomic bids they are based on.

Definition 2.1. [11] An atomic bid is a pair (S, p), indicating that the bidder values any superset of S (including S) at p and everything else at 0. Formally,

$$v(X) = \begin{cases} p & \text{if } S \subseteq X \subseteq G \\ 0 & \text{otherwise} \end{cases}$$

If we restrict the bidder to one atomic bid, then this precisely describes the *single-minded* class of valuations. An easy way to extend this concept to achieve full expressivity is to allow for chains of atomic bids connected with the XOR operator. This describes the *XOR bidding language*, consisting of *XOR bids*.

Definition 2.2. [11] An *XOR bid* has form

$$\bigoplus_{a=1}^{A} (S_a, p_a),$$

where the value for a subset of G is given by

$$v(S) = \max_{a : S_a \subseteq S} p_a.$$

Since we could assign a separate atomic bid for each subset, the XOR bidding language is fully expressive and *polynomially interpretable*, meaning that the value of a set can be queried in polynomial time. However, the language is not very concise—even additive valuations require exponential space to store. Therefore, Sandholm introduces the broader *OR-of-XOR* bidding language, where

bids are represented as ORs of XOR clauses:

$$c_1 \vee c_2 \vee \cdots \vee c_a,$$

where each c_i is an XOR clause. Goods may appear in multiple clauses, so the valuation of a subset S is the maximum value obtained by any assignment of each $j \in S$ to some appearance of j in the OR-of-XOR expression. If multiple atomic bids within the same clause are satisfied, the value is a max over the atomic bids; if multiple clauses are satisfied, the value is a sum over all values obtained by each clause.

If we were to restrict ourselves to just ORs of atomic bids, we would lose expressivity, as substitutes are impossible to express. Therefore, the *OR-of-XOR* bidding language takes the best of both worlds from OR and XOR. However, we will find that in other circumstances the language can be quite limiting.

$\mathcal{L}_{GB}$ and TBBL

In 2001, Boutilier and Hoos[3] constructed the $\mathcal{L}_{GB}$ language by extending the OR/XOR languages in several ways. First, they add the boolean operators $\wedge$ and allow any tree structure of boolean operators, rather than restricting to clausal normal form. Second, they allow for values at any of the nodes of the tree. For example, $((a, 2) \wedge (b, 2), 1)$ describes a complementary relationship with a "bonus" of 1 for obtaining both items ($v(a) = v(b) = 2, v(\{a, b\}) = 3$).

While the $(\cdot \oplus \cdot)$ operator (without an attached value) can still be used to represent complete substitutability as in the OR/XOR langauges, we can also represent partial substitutability by use of the $(\cdot \vee \cdot, p)$ operator. For example, $((a, 1) \vee (b, 2), 10)$ indicates substitutability with a slight preference for b over a ($v(a) = 11, v(b) = v(\{a, b\}) = 12$).

Boutilier and Hoos give the following example of a valuation function which has a compact representation in $\mathcal{L}_{GB}$, but would need to be exponential in OR-of-XOR:[2]

Example 2.3. [3] Consider the bid:

$$\langle m \wedge r_1, p_1 \rangle \vee \langle m \wedge r_2, p_2 \rangle \vee \cdots \vee \langle m \wedge r_k, p_k \rangle$$

Here, m can be interpreted as a machine, which is necessary to satisfy any of the clauses, and $r_1, \ldots, r_k$ are interpreted as raw materials, each of which can be used in conjunction with the machine to generate some payoff.

In 2005, a new bidding language called TBBL[3] was introduced Cavallo et al[4]. Similar to $\mathcal{L}_{GB}$, the language is similarly built on boolean operators in a tree-like structure, but there are two main differences:

- When evaluating a set, the bidder receives value for *all* satisfied clauses—there is no constraint

[2] Or even XOR-of-OR, which is more general.
[3] Tree-based Bidding Language.

than an item cannot be used for more than one.

- It is possible to specify that "k out of n" clauses need to be satisfied rather than just one (OR) or all (AND).

The second condition in particular makes it easy to construct compact TBBL bids which require exponential space in $\mathcal{L}_{GB}$ (and thereby all of the other boolean-based bidding languages we've discuessed). Cavallo et al give the following real-world example of such a valuation:

> An airline has interest in a set of takeoff times at an airport that regulates the number of takeoffs and landings by allocating "time-slots" to different airlines. There are five evening slots $(E_1, E_2, E_3, E_4, E_5)$ and one morning slot (M) on the market, and the airline has a business plan that requires acquisition of the morning slot and 2 or 3 of the evening slots, but no more, where the precise value of the allocation depends on which evening slots are received. The evening slots yield value 1, 2, 3, 4, and 5 respectively to the airline if two or three are acquired along with the morning slot, and there is a bonus of 10 for achieving such a business plan.

Cavallo et al use what the call the "interval-choose" operator IC_2^3 to denote that exactly 2 or 3 of the 5 evening slots must be satisfied. The resulting TBBL bid is

$$\langle M \wedge \langle IC_2^3 : \langle E_1, 1\rangle, \langle E_2, 2\rangle, \langle E_3, 3\rangle, \langle E_4, 4\rangle, \langle E_5, 5\rangle\rangle, 10\rangle.$$

Since there is no compact way of simulating the IC operator with boolean operators, an $\mathcal{L}_{GB}$ representation of this valuation requires enumerating all subsets of the evening slots of size 2 or 3. Therefore, an IC clause of size k can take $\Omega(2^k)$ space to represent in $\mathcal{L}_{GB}$.

Hypergraphs and $\mathcal{MPH}$

As far as we know, the idea of using hypergraphs to represent valuation functions was born in [5] in 2005. Recall that we use a vertex weights to represent item values and hyperedge weights to represent marginal gains or losses from obtaining a set in full. The resulting value of a subset S is given as follows:

$$v(S) = \sum_{U \subseteq S} h(U),$$

where $w(\varnothing) = 0$ and $w(\{j\}) = v(j)$ for every $j \in G$. In the scenarios we study, the hypergraph will be sparse enough that the number of nonzero-weighted hyperedges is polynomial in m. For a more formal treatment of hypergraphs and their corresponding valuation functions, see section 3.

While hypergraphs are fully expressive, there are some simple valuations for which their representation is exponential. For example, unit demand valuations generally require hypergraphs of exponential size (i.e. an exponential number of nonzero-weighted hyperedges) to store. However, if we use *multiple* hypergraphs to store a single valuation, we can solve this problem. In 2014, Feige et al[7] define the Maximum over Positive Hypergraphs representation, in which a single valuation

function is represented by multiple hypergraphs, joined by a max operator. Formally,

$$v(S) = \max_a v_a(S),$$

where v_a is the valuation function induced by the ath hypergraph. Now unit demand valuation is easy to represent: for each item j with nonzero value p, construct a hypergraph in which j has weight p.

If a single hypergraph cannot even represent unit demand valuations concisely, why then, do we opt to use a single hypergraph in this paper? For one, we believe it is a simpler representation. When we attempt to solve the winner determination problem, we will only need to consider n hypergraphs, each of which completely characterize a single bidder's valuation function.

More importantly, we opt to study valuation functions which do not fall into the well-established classes. These well-known classes include (in increasing order of generality): additive, unit-demand, gross substitutes, submodular, and monotone valuations. Though hypergraphs may not be able to represent unit-demand valuations, our winner determination algorithms apply to valuation functions which are not even monotone[4]. Thus, the valuation functions for which our winner determination algorithms work are lesser-studied and little explored. However, we believe they are practical, as we will justify with example use cases in section 3.

2.2 Winner Determination

In order to hope to achieve polynomial-time winner determination for any of the bidding languages, we need to place some *restriction* on the language. We will review two restrictions—one on the OR-of-XOR language and one on hypergraphs—which have lead several authors to achieve polynomial-time winner determination.

Recall that if we restrict bidders to exactly one atomic bid each, winner determination is still NP-hard. Therefore, we will need a new notion of atomic bid if we hope to achieve poly-time winner determination. Consider the following:

Definition 2.4. [8] A *singleton* is an atomic bid (S, p) where $|S| = 1$.

It turns out that if we restrict OR-of-XOR bids to singletons, we are able to solve the winner determination problem in polynomial time by bipartite maximum-weight matching.[10] This class is called OXS (OR-of-XOR of Singletons) and is a strict superset of the additive and unit demand valuation classes, and a strict subset of the gross substitutes class.[8] Flipping the boolean operators, we can consider the XOR-of-OR of singletons class, or XOS, which is more general than OXS. In fact, it is a strict superset of the submodular valuation class. While there is no known polynomial-time winner determination algorithm for the XOS class, Dobzinski et al[6] introduce a polynomial-time 2-approximation.

[4] $\mathcal{MPH}$ cannot represent non-monotone valuations.

The authors of the original hypergraph paper[5] defined hypergraphs generally, but focused primarily on those for which hyperedges connect to at most two vertices. We call this a *rank-2* restriction. This enforces that bids are of polynomial size. However, they show that winner determination is NP-hard in the general case. However, they are able to achieve polynomial-time winner determination with the following restriction on rank-2 hypergraphs:[5]

> G is a forest, where G is the graph of all vertices (items) and all edges between items such that at least one bidder places a nonzero value on the edge.

For each tree in the forest, they pick a root and compute the welfare-maximizing allocation dynamically from leaves to the root.

In 2012, Abraham et al[1] were able to make the looser restriction that G must be a planar graph[5] and obtain a $(1+\epsilon)$-approximation of the optimal social welfare. Further they are able to give a polynomial-time r-approximations for hypergraphs restricted to rank r, which they show is tight. Similarly, Feige et al[7] obtain a polynomial-time $k+1$ approximation of the optimal social welfare when each of the hypergraphs describing a valuation are restricted to rank k.

Given the tightness of Abraham et al's approximation, it is clear that we will need to restrict hypergraphs in some other way in order to achieve polynomial-time winner determination. In the next section, we will introduce the "restriction on overlapping" which will be the focus of this paper.

3 Our Model

We formally introduce the hypergraph representation of valuations and provide an illustrative example. Then we introduce the notion of *overlapping* and discuss several ways we could choose to restrict it. For each restriction, we give examples—both simple and real-world—and we state our results.

3.1 Hypergraph Valuations

Hypergraphs

A *hypergraph* is a generalization of a graph in which edges are permitted to connect to any subset of the vertices, not necessarily of size two. These generalized edges are referred to as *hyperedges*. A *weighted hypergraph* is a hypergraph with values on each hyperedge. (Note that values on vertices are redundant because a hyperedge can connect to a lone vertex.)

We use the following notation:

- A hypergraph H has vertices $V(H)$ and hyperedges $E(H)$.
- For a hyperedge $h \in E(H)$, we use $S_h \subseteq V(H)$ to refer to the set of vertices incident to h.

[5] That is, it does not contain a 5-clique or 3×3 bipartite graph.

- For a hyperedge $h \in E(H)$, we use $w_h \in \mathbb{R}$ to refer to the weight on h. Alternatively, we may use $w(S)$ to refer to the weight on the hyperedge associated with $S \subseteq V(H)$ where w is understood to be associated with H.

The Auction Setting

We use the following notation to describe the auction setting:

- A set N of bidders with $|N| = n$.
- A set G of goods with $|G| = m$.
- For each bidder i, a type space $T_i \subseteq \mathbb{R}^+$ from which i's valuations can be drawn.
- For each bidder i, a valuation function $v_i : G \to T_i$, where $v_i(j)$ denotes bidder i's valuation for the good $j \in G$. Additionally, we may use $v_i : 2^G \to T_i$, where $v_i(S)$ denotes bidder i's value for obtaining the set $S \subseteq G$.

Hypergraph Valuations

We use a hypergraph H to describe a single bidder's valuation function v as follows:[5]

- Let there be m vertices in $V(H)$ for each of the m goods.
- If h is incident to a lone vertex corresponding to item $j \in G$, define $w(S_h) = v(j)$.
- If h is incident to exactly two vertices/goods j and j', then $w(S_h) = v(\{j, j'\}) - w(\{j\}) - w(\{j'\})$.
- If h is incident to a set $S_h \subseteq G$, then $w(S_h) = v(S_h) - \sum_{S \subsetneq S_h} w(S)$.

This way, $w(S)$ describes the marginal gain or loss from obtaining S as a bundle. Rearranging terms, we find that for some set $S \subseteq G$,

$$v(S) = \sum_{U \subsetneq S} w(U).$$

While this may look exponential, we will typically consider sparse graphs, in which most of the hyperedges will have weight 0 and so do not need to be considered. Therefore, if $w(h) = 0$ for some h in the above definition, we say that $h \notin E(H)$. Consider the additive valuation class, for instance. In this class, the value of a set $S \subseteq G$ is given by

$$v(S) = \sum_{j \in S} v(S).$$

This corresponds to a hypergrpah in which the only nonzero hyperedges are on isolated vertices, whose values are equal to the corresponding item values. Therefore, $|E(H)| = m$, where H describes

an additive valuation. We analyze other common valuation classes such as unit demand and monotone in the Appendix.

We demonstrate hypergraph representation with a simple example, illustrated in Figure 1, in which three items exhibit complementary relationships. We imagine an agent who would like to purchase a single kayak and paddle. Therefore the kayaks and paddles do not have much value as standalone items (2 for each kayak and 1 for the paddle), but the combination of a kayak and paddle is worth 10 or 11, depending on the color of the kayak.

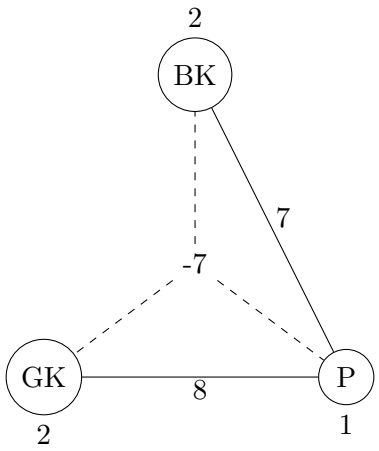


Figure 1: Hypergraph example. K = kayak, P = paddle, B = blue, G = green.

3.2 Allocations

Since the goal of this paper is to find algorithms which compute the welfare-maximizing allocation, we need to specify exactly what the return type of these algorithms will be. We take this opportunity to formalize the notion of an 'allocation'.

We choose to represent an allocation as a function $\mathcal{A} : G \to N$ that assigns each good to a bidder. However, we can also view a function $f : X \to Y$ as a subset $f \subseteq X \times Y$ with the constraint that for every $x \in X$, there is exactly one $y \in Y$ such that $(x, y) \in f$. Therefore, the winner determination algorithms we propose in later sections will return a subset of $G \times N$. We also find it useful to rank potential allocations by the welfare they generate. With this in mind, we introduce the following definitions

Definition 3.1. An *allocation* $\mathcal{A}$ is a function $\mathcal{A} : G \to N$. Let $\mathcal{A}^{-1}$ be the inverse of $\mathcal{A}$: $\mathcal{A}^{-1}(i) = \{j \in G \mid \mathcal{A}(j) = i\}$. The set of all allocations is given an implicit partial ordering by the welfare function wel$(\cdot)$ (defined below).

Definition 3.2. The *welfare* of an allocation $\mathcal{A}$ is given by the following:

$$\text{wel}(\mathcal{A}) = \sum_{i \in N} v_i \left(\mathcal{A}^{-1}(i) \right).$$

Since we take wel$(\cdot)$ to be an implicit partial ordering on allocations, we have that $\mathcal{A}_1 > \mathcal{A}_2$ if

$$\sum_{i \in N} v_i\left(\mathcal{A}_1^{-1}(i))\right) > \sum_{i \in N} v_i\left(\mathcal{A}_2^{-1}(i)\right).$$

In this case, we have that $\max(\mathcal{A}_1, \mathcal{A}_2) = \mathcal{A}_1$.

Sometimes we find it useful to consider an allocation of some subset $S \subseteq G$. Therefore, we give a similar definition for $\mathcal{A}^S$, which only allocates the goods in $S \subseteq G$.

Definition 3.3. A *partial allocation* $\mathcal{A}^S$ is a function $\mathcal{A}^S : S \to N$, where $S \subseteq G$. We define $(\mathcal{A}^S)^{-1} : N \to 2^G$ to be the inverse of $\mathcal{A}^S$. Partial allocations are given an implicit partial ordering as follows: $\mathcal{A}_1^S > \mathcal{A}_2^S$ if

$$\sum_{i \in N} v_i\left((\mathcal{A}_1^S)^{-1}(i)\right) > \sum_{i \in N} v_i\left((\mathcal{A}_1^S)^{-1}(i)\right).$$

3.3 Restrictions on Multiple Hypergraphs

We will soon discuss restrictions on hypergraphs that will allow for polynomial-time winner determination. But before we do so, we need to discuss what it means to impose a restriction on *multiple* hypergraphs—those of each of the n bidders. Consider the following restriction on a hypergraph:

Only one (nonzero-valued) hyperedge is permitted on the hypergraph.

This is an extremely stringent restriction—one would hope that we can solve winner determination for considerably more general cases. However, if we enforce that each bidder's individual hypergraph has this restriction, this is equivalent to the single-minded class of valuations; as we saw in section 2, winner determination is NP-hard.

Instead of imposing the same restriction on each bidder's hypergraph, we construct a single hypergraph H_{union} out of each of the n individual ones.

Definition 3.4. Suppose each bidder i has a hypergraph H_i. A set S has a corresponding hyperedge in $E(H_{\text{union}})$ if and only if there exists some bidder i such that $w(i) \neq 0$.

The restrictions we study, which we will introduce shortly, always concern this "combined hypergraph" H_{union}, even when not explicitly stated. Recasting the single-hyperedge restriction in these terms, we obtain the following:

Pick a set S. Each bidder's hypergraph may only contain one hyperedge, which corresponds to S.

Now the winner determination problem is easy: give S in full to the bidder who values it the most. We now introduce some looser restrictions on the graph H_{union} in search of reasonably expressive valuation classes for which we can solve winner determination.

Enforcing a restrictions on the "combined hypergraph" is in general much stricter than enforcing the same restriction on each bidder's individual graph. However, we believe this is a reasonable

assumption to make in the real world. In many scenarios, bidders will possess similar dependencies between items. For example, we expect that most bidders' hypergraphs will contain an edge between two competing brands of lawnmowers—most individuals only need one—whereas a lawnmower and a novel have no relationship to each other, and therefore we would not expect a hyperedge between them on any bidder's hypergraph. The combined hypergraph encapsulates all dependencies that bidders *might* have (it is of course fine if some bidder's hypergraphs look much sparser than others), which we believe is sufficient for most real-world scenarios.

In the next sections, we study restrictions on hypergraphs. Whenever we impose one of these restrictions on multiple hypergraphs for the purpose of computing the optimal allocation, we always mean that the combined hypergraph has this restriction.

3.4 Overlap

Given $S, S' \subseteq G$, we say that S and S' *intersect* if $S \cap S' \neq \varnothing$. While we could choose to limit intersection, we find that allowing containment, i.e. $S \subseteq S'$ or $S' \subseteq S$, does not make the winner determination problem harder. Therefore, we will use the following notion of 'overlap', which adds the additional constraint that $S \not\subseteq S'$ and $S' \not\subseteq S$.

Definition 3.5. We say that hyperedges h and h' *overlap* (or *are overlapping*) if their corresponding sets S_h and $S_{h'}$ have the following property:

$$S_h \cap S_{h'} \neq \varnothing, S_h \not\subseteq S_{h'}, \text{ and } S_{h'} \not\subseteq S_h.$$

Alternatively, we may apply this term to sets directly, rather than hyperedges. Then $S, S' \subset G$ overlap if

$$S \cap S' \neq \varnothing, S \not\subseteq S', \text{ and } S' \not\subseteq S.$$

The first restriction we study—and the most stringent—is to disallow overlap entirely. With this restriction, we are able to solve the winner determination problem in polynomial time by using a dynamic program on the graph. The algorithm is given in section 4.

3.5 Dense Overlap and Broad Overlap

Since we are able to solve winner determination in polynomial time for the previous scenario, we search for ways to allow *some* overlapping, but not so much that we lose poly-time winner determination.

One can imagine two separate notions of overlapping that might complicate algorithms on the hypergraph. We call the first kind *dense overlap*. Imagine a Venn Diagram with four circles, or five or six. Even with a small number of circles, there becomes a large mess of interconnected regions. We measure this by considering some item $j \in G$ and counting the number of overlapping hyperedges that contain j. If, for example, we made the restriction that no item j can be contained inside three overlapping hyperedges, then this immediately rules out the classic 3-circle Venn diagram.

The other notion, which we call *broad overlap*, is the hypergraph analogue of a highly connected graph. Rather than isolating an item and counting the number of hyperedges that contain it, we isolate a hyperedge and count the number of other hyperedges that overlap with it.

How can we formalize these notions? The fact that we do not incorporate containment into our definition of overlapping complicates matters slightly. For dense overlap, for instance, it is not sufficient to just count the number of hyperedges containing a particular item $j \in G$, since those hyperedges could be contained inside each other and therefore not be overlapping. Therefore, we introduce a method of analyzing overlap by viewing hyperedges as nodes in a new graph, and overlap between hyperedges as edges.

For a hypergraph H, we construct a new (ordinary) graph G_H in which each hyperedge is represented by a node in G_H, and the vertices corresponding to hyperedges h_1 and h_2 are joined by an edge if h_1 and h_2 overlap.

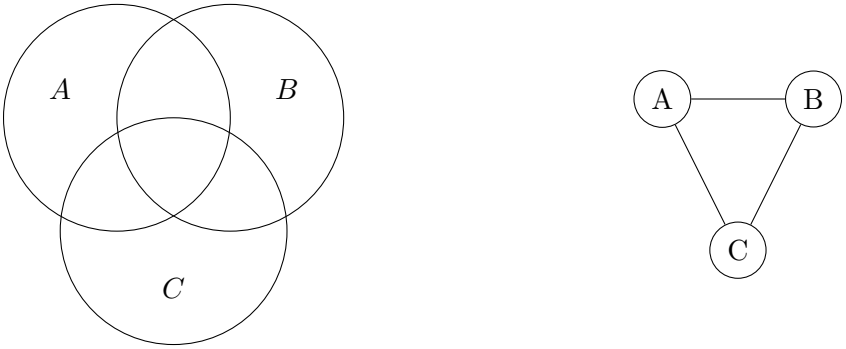


Figure 2: A Venn diagram intersection and its corresponding G_H graph.

The issue with this representation is that it doesn't capture the triple overlap that occurs in the Venn diagram in Figure 2. For instance, the hypergraph depicted in Figure 3 has the same graph G_H, although we would like to say that it has a lesser degree of overlapping.

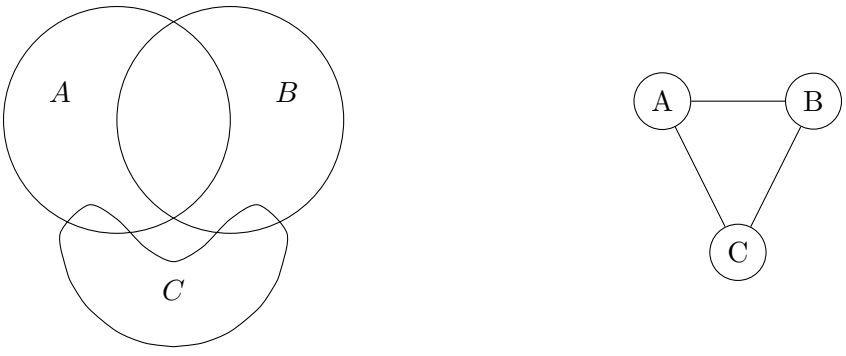


Figure 3: Another hypergraph with the same G_H graph.

The key difference is that in Figure 2, there are items which lie at the intersection of three overlapping hyperedges, whereas in Figure 3. Therefore, we can pick some item $j \in G$ and restrict the graph G_H to only represent hyperedges which contain the good j. We call the resulting subgraph

$G_H|_j$. If we pick some j in the shaded region in Figure 4, then we end up with the same graph because each of the hyperedges contains j, but if we pick some j in the shaded region in Figure 5, we end up with a simpler graph $G_H|_j$, also shown in Figure 5.

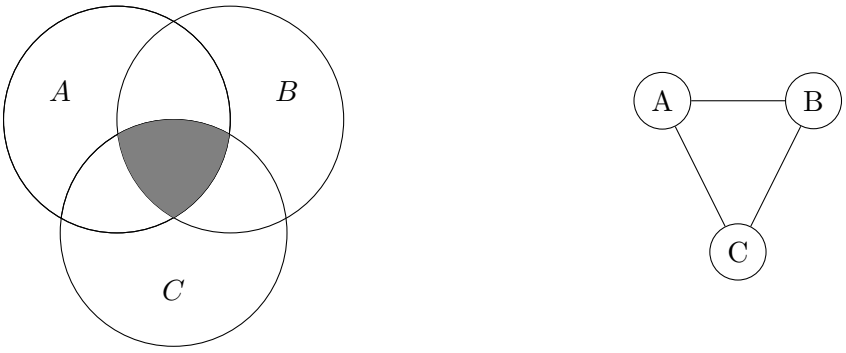


Figure 4: Picking a j from the center yields the same graph on the right.

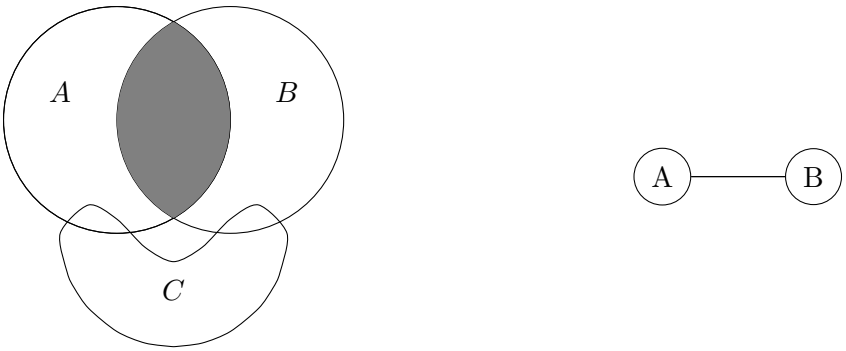


Figure 5: No matter where we pick j, $G_H|_j$ will have at most two connected nodes.

Our restrictions will take the form: $\Delta(G_H) \leq k$ (restriction on breadth) and $\Delta(G_H|_j) \leq k$ for all $j \in G$ (restriction on density), where $\Delta(G)$ denotes the maximal degree of any vertex. Observe that if we restrict $\Delta(G_H) \leq k$, then it is impossible for any item to lie at the intersection of more than k overlapping hyperedges, as that would imply that each one of those hyperedges overlap with each other. (Another way of seeing this is that if some subgraph of G violates the restriction then G does as well.) Therefore, $\Delta(G_H) \leq k$ is a tighter restriction, so our hopes are higher for finding polynomial winner determination for breadth restriction. Indeed, we show in section 5 that even if we restrict $\Delta(G_H) \leq 1$, the winner determination problem is NP-hard. However in the section after that, we are able to show that restricting $\Delta(G_H) \leq 2$, still yields a polynomial time algorithm.

4 No Overlap

First, we tackle the problem of finding a welfare-maximizing allocation in the scenario where no two hyperedges overlap. That is for any S, S' with nonzero hyperedge weights,

$$S \subset S', S' \subset S, \text{ or } S \cap S' = \varnothing.$$

The algorithm views the containments as a tree and uses a rootward dynamic program to achieve polynomial running time.

4.1 Algorithm

We view the hypergraph as a tree, in which each node represents a hyperedge h (with nonzero weight for some bidder) with the following rule: if h' is a descendant of h, then $S_{h'} \subset S_h$. We use a dummy hyperedge r (possibly of weight $\mathbf{0}$) to use as the root of the tree. We will write an allocation function $\mathcal{A}_{\text{OPT}}^{S_h}$ which returns the optimal allocation of the items in S_h. Therefore, to compute the welfare-maximizing allocation, we call $\mathcal{A}_{\text{OPT}}^{S_r}$.

The algorithm consists of several recurrence relations. For now, we state them with brief explanations, which will conclude the description of the algorithm. Later we will prove that each of these behave as intended, and that we can use dynamic programming to run $\mathcal{A}_{\text{OPT}}^{S_r}$ in polynomial time.

We begin by introducing the following notation:

- $i^{(1)}(S) = \min \left\{ i \mid i \in \arg\max_{i' \in N} v_{i'}(S) \right\}$
- $i^{(2)}(S) = \min \left\{ i \mid i \in \arg\max_{i' \in N \setminus \{i^{(1)}(S)\}} v_{i'}(S) \right\}$

$i^{(1)}(S)$ is simply the bidder i who values S the most, breaking ties by favoring smaller values of i. $i^{(2)}(S)$ is the bidder who values S second-most.

We now proceed to the recurrences.

$\mathcal{A}_{\text{OPT}}^{S_h}$ returns the best, or welfare-maximizing allocation for all goods in S_h. $\mathcal{A}_{\text{OPT}}^{S_h}$ is given by the following:

$$\mathcal{A}_{\text{OPT}}^{S_h} = \max(\mathcal{A}_{\text{B}}^{S_h}, \mathcal{A}_{\text{ALT}}^{S_h}).$$

$\mathcal{A}_{\text{B}}^{S_h}$ returns the welfare-maximizing allocation if we enforce that all goods in S_h are allocated to the same bidder. (We call such allocations "bundle allocations".)

$$\mathcal{A}_{\text{B}}^{S_h} = S_h \times \{i^{(1)}(S_h)\}.$$

$\mathcal{A}_{\text{B}'}^{S_h}$ returns the *second-best* allocation for all goods in S_h.

$$\mathcal{A}_{\text{B}'}^{S_h} = S_h \times \{i^{(2)}(S_h)\}.$$

$\mathcal{A}_{\text{ALT}}^{S_h}$ returns the best allocation for all goods in S_h that is *different than* $\mathcal{A}_{\text{B}}^{S_h}$:

$$\mathcal{A}_{\text{ALT}}^{S_h} = \begin{cases} \max(\mathcal{A}_{\text{B}'}^{S_h}, \mathcal{A}_{\text{GS}}^{S_h}) & \mathcal{A}_{\text{G}}^{S_h} = \mathcal{A}_{\text{B}}^{S_h} \\ \max(\mathcal{A}_{\text{B}'}^{S_h}, \mathcal{A}_{\text{G}}^{S_h}, \mathcal{A}_{\text{GS}}^{S_h}) & \text{otherwise} \end{cases}$$

$\mathcal{A}_{\text{G}}^{S_h}$ returns the best allocation ignoring the weight on h (supposing $\mathbf{w_h} = \mathbf{0}$).[6]

$$\mathcal{A}_{\text{G}}^{S_h} = \bigcup_{h' \text{ child of } h} \mathcal{A}_{\text{OPT}}^{S_{h'}}$$

If $\mathcal{A}_{\text{G}}^{S_h}$ allocates all items in S_h to a single bidder and $|S_h| > 1$, then $\mathcal{A}_{\text{GS}}^{S_h}$ finds the optimal allocation in which S_h is split among at least two bidders. Otherwise, $\mathcal{A}_{\text{GS}}^{S_h}$ is the empty allocation.

$$\mathcal{A}_{\text{GS}}^{S_h} = \begin{cases} \left(\mathcal{A}_{\text{G}}^{S_h} \setminus \mathcal{A}_{\text{B}}^{S_{h^*}}\right) \cup \mathcal{A}_{\text{ALT}}^{S_{h^*}} & |S_h| > 1 \text{ and } \mathcal{A}_{\text{G}}^{S_h} = S_h \times i \text{ for some } i \\ \varnothing & \text{otherwise,} \end{cases}$$

where

$$h^* \in \underset{h' \text{ child of } h}{\arg\min} \left(\text{wel}(\mathcal{A}_{\text{B}}^{S_{h'}}) - \text{wel}(\mathcal{A}_{\text{ALT}}^{S_{h'}})\right).$$

4.2 Correctness

We prove that $\mathcal{A}_{\text{OPT}}^{S_h}$ accurately finds the optimal allocation for the set of items S_h to the set of bidders N. To do this, we inductively prove the correctness of the recurrences

$$\mathcal{A}_{\text{OPT}}, \mathcal{A}_{\text{B}}, \mathcal{A}_{\text{B}'}, \mathcal{A}_{\text{ALT}}, \mathcal{A}_{\text{G}}, \mathcal{A}_{\text{GS}}. \tag{1}$$

We first prove the base cases: assume $|S_h| = 1$, that is, h connects to a lone vertex.

- $i^{(1)}(\{j\})$ and $i^{(2)}(\{j\})$ give the bidder who values j the most, so $\mathcal{A}_{\text{B}}^{S_h}$ and $\mathcal{A}_{\text{B}'}^{S_h}$ allocate j to the bidders who value j most and second-most, as expected.
- $\mathcal{A}_{\text{G}}^{S_h}$ and $\mathcal{A}_{\text{G}}^{S_h}$ both return the empty allocation, as expected.
- $\mathcal{A}_{\text{ALT}}^{S_h} = \mathcal{A}_{\text{B}'}^{S_h}$, since $\mathcal{A}_{\text{G}}^{S_h}$ and $\mathcal{A}_{\text{G}}^{S_h}$ generate no welfare. Then $\mathcal{A}_{\text{ALT}}^{S_h}$ gives the good to bidder who values it second-most, which is indeed the second-best allocation.

[6] If h corresponds to a single, "ignoring the weight on h" means that no welfare is achievable, so we return the empty allocation.

- $\mathcal{A}_{\text{OPT}}^{S_h} = \max(\mathcal{A}_{\text{B}}^{S_h}, \mathcal{A}_{\text{B}'}^{S_h})$, and $\mathcal{A}_{\text{B}}^{S_h}$ gives the item to the bidder who values it the most, which is optimal. Therefore, $\mathcal{A}_{\text{OPT}}^{S_h}$ finds the optimal allocation in the case that $|S_h| = 1$.

Now, assume all of the above allocations match behave as outlined in the description of the algorithm for every child h' of h. We will show that each of the functions return the correct allocation when applied to h.

$\mathcal{A}_{\text{B}}^{S_h}$ and $\mathcal{A}_{\text{B}'}^{S_h}$ simply give all items in S_h to $i^{(1)}(S_h)$ and $i^{(2)}(S_h)$, which are defined as the bidder who wants the set the most and second-most. Therefore, $\mathcal{A}_{\text{B}}^{S_h}$ and $\mathcal{A}_{\text{B}'}^{S_h}$ are the best and second-best "bundle allocations".

$\mathcal{A}_{\text{G}}^{S_h}$ is also quite intuitive. If we ignore the weight on h, then the allocation of $S_{h'}$ for some child h' of h has no impact on the welfare of any allocation of goods in $S_h \setminus S_{h'}$. Therefore, we maximize welfare in S_h by maximizing welfare piecewise for each set $S_{h'}$, where h' is a child of h. This is performed by the allocation $\mathcal{A}_{\text{OPT}}$, so assuming $\mathcal{A}_{\text{OPT}}$ correctly returns the optimal allocation for each $S_{h'}$, $\mathcal{A}_{\text{G}}^{S_h}$ correctly returns the best allocation ignoring $\mathbf{w_h}$.

The description of $\mathcal{A}_{\text{GS}}^{S_h}$ is conditioned on the allocation $\mathcal{A}_{\text{G}}^{S_h}$. Assume that for some bidder i, $\mathcal{A}_{\text{G}}^{S_h} = S_h \times \{i\}$. Since we are maximizing over allocations which split up S_h to multiple bidders, the weight of h will not be awarded. Thus, as before, the allocation of $S_{h'}$ for some child h' of h has no impact on the welfare of any allocation of goods in $S_h \setminus S_{h'}$. We know that maximizing piecewise will result in $S_h \times \{i\}$, so we maximize piecewise for all but one child h'. We allocate h' using the second-best allocation and we select h' so as to minimize the loss in welfare. This successfully maximizes welfare given the constraint that i cannot receive the entire bundle S_h.

$\mathcal{A}_{\text{ALT}}^{S_h}$ is the most significant piece of the algorithm conceptually. The claim is that $\mathcal{A}_{\text{ALT}}^{S_h}$ finds the best allocation among all allocations different from $\mathcal{A}_{\text{B}}^{S_h}$—for now let's call this allocation OPT$'$. Either OPT$'$ allocates all items to the same bidder, or it allocates to multiple bidders. In the first case, OPT$' = \mathcal{A}_{\text{B}'}^{S_h}$ (it cannot be $\mathcal{A}_{\text{B}}^{S_h}$ because it is different from $\mathcal{A}_{\text{B}}^{S_h}$ by definition). If OPT$'$ allocates to multiple bidders, then the allocation of $S_{h'}$ for some child h' of h has no impact on the allocations of $S_h \setminus S_{h'}$. In this case we allocate each $S_{h'}$ optimally—this is precisely what $\mathcal{A}_{\text{G}}^{S_h}$ does. In the event that $\mathcal{A}_{\text{G}}^{S_h}$ allocates all items to the same bidder i, we have shown that $\mathcal{A}_{\text{GS}}^{S_h}$ gives the optimal allocation in which S_h is split up among two or more bidders. Therefore OPT$' = \mathcal{A}_{\text{GS}}^{S_h}$. We have shown that OPT$'$ will be one of $\mathcal{A}_{\text{B}'}^{S_h}$, $\mathcal{A}_{\text{G}}^{S_h}$, or $\mathcal{A}_{\text{GS}}^{S_h}$. If $\mathcal{A}_{\text{G}}^{S_h} = \mathcal{A}_{\text{B}}^{S_h}$, then $\mathcal{A}_{\text{ALT}}^{S_h}$ correct returns the max of $\mathcal{A}_{\text{B}'}^{S_h}$ and $\mathcal{A}_{\text{GS}}^{S_h}$ because each of these are guaranteed to be different from $\mathcal{A}_{\text{B}}^{S_h}$. Otherwise, $\mathcal{A}_{\text{ALT}}^{S_h}$ will take a max over $\mathcal{A}_{\text{B}'}^{S_h}$, $\mathcal{A}_{\text{G}}^{S_h}$, and $\mathcal{A}_{\text{GS}}^{S_h}$, thereby finding OPT$'$.

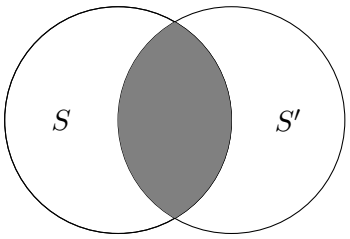
$\mathcal{A}_{\text{OPT}}^{S_h}$ takes the better of $\mathcal{A}_{\text{B}}^{S_h}$ and $\mathcal{A}_{\text{ALT}}^{S_h}$. Since we showed that $\mathcal{A}_{\text{B}}^{S_h}$ gives the best "bundled allocation" and $\mathcal{A}_{\text{ALT}}^{S_h}$ gives the best allocation different from $\mathcal{A}_{\text{B}}^{S_h}$, the optimal allocation will be either $\mathcal{A}_{\text{B}}^{S_h}$ and $\mathcal{A}_{\text{ALT}}^{S_h}$. (If the allocation gives all items to one bidder, then it is $\mathcal{A}_{\text{B}}^{S_h}$; if it splits the items

up, then $\mathcal{A}_{\text{ALT}}^{S_h}$ finds the best allocation in the larger space of "allocations that aren't $\mathcal{A}_{\text{B}}^{S_h}$", so $\mathcal{A}_{\text{B}}^{S_h}$ will be the optimal allocation.)

Since there are polynomially many hyperedges and each recurrence requires polynomial computations, the algorithm can be run in polynomial time by memoizing the values in each of the recurrences.

5 Density-Restricted Overlap

We study the class of hypergraph representations w for which the subgraphs $G_w|_j$ satisfy $\Delta(G_w|_j) \leq 1$ for all items $j \in G$. To make this restriction more intuitive, we provide the following example. Consider two overlapping hyperedges w and w' with corresponding sets S and S', respectively:



If we add a third hyperedge, it can either lie entirely inside the shaded region or entirely outside. However, it can overlap with $S \setminus S'$ and/or $S' \setminus S$.

Therefore, we can view the set of hyperedges as a set of connected components $C_1, C_2, \ldots, C_k$ with each pair either being entirely disjoint or one lying entirely within some region of the other.

Note that while the overlapping seems "simple" in some sense because there are no triple-intersections, there are no restrictions on the graph G_h. For instance, although it is not possible to draw a diagram of this in the plane, each connected component can potentially correspond to a complete graph in the G_h representation. This freedom in G_h turns out to cause the winner determination to be NP-hard even with this seemingly stringent restriction on the overlaps.

Theorem 5.1. *Winner determination is NP-hard for the class of hypergraph representations h for which $\Delta(G_h|_j) \leq 1$ for all $j \in G$.*

Proof. We prove NP-hardness via a reduction from 3-Colorability. Consider an arbitrary instance $\langle G \rangle$ of 3-Colorability, we produce a hypergraph representation h of a valuation function as follows:

- Let $d = |V(G)|$. For each vertex in $V(G)$ we construct a set of $d-1$ items, each valued at 1 by *all* bidders. Let this set have weight 1.
- For every edge $(u, v) \in E(G)$ with u and v corresponding to some sets S_u and S_v with nonzero hyperedge weight, pick $a \in S_u$, $b \in S_v$, and let $\{a, b\}$ have weight -1. Specifically, pick items a and b that were not contained in any hyperedge-sets besides S_u and S_v[7].

[7]Since there are $d-1$ vertices in each of these sets, this is always possible to do.

Let there be 3 bidders, each of whom has this exact hypergraph h. Given an arbitrary graph G, we claim that $\langle G \rangle \in$ 3-Colorability if and only if the maximum possible welfare given the corresponding hypergraph representations is at least d^2.

Let $\langle G \rangle \in$ 3-Colorability. Then there exists a way to label the nodes in G with one of $\{1, 2, 3\}$ such that no two adjacent nodes have the same label. Consider the following allocation: For each node $v \in G$ corresponding to a hyperedge-set S_v with some label $i \in \{1, 2, 3\}$, give all items S to bidder i. This allocation is feasible since none of the hyperedge-sets corresponding to vertices in $V(G)$ overlap, and achieves welfare d^2: 1 for each of the $d-1$ items in each of the d hyperedge-sets ($(d-1) \cdot d$ total) plus 1 for each of the d fully-awarded hyperedge-sets S_v corresponding to $v \in V(G)$ (d total). This sums to d^2. We know that none of the negative-weighted hyperedges are awarded because they only exist of pairs of items from adjacent vertices in G, which are not awarded to the same person because this would violate the 3-coloring. Therefore, the total welfare of this allocation is d^2.

Suppose it is possible to achieve welfare d^2. This means that all items are awarded and all positive hyperedges (corresponding to vertices in $V(G)$) are awarded and no negative hyperedges (corresponding to edges in $E(G)$) are awarded. Consider the following labelling of vertices in G: for each hyperedge-set S_v corresponding to some $v \in V(G)$ for which all items were awarded to some bidder i, label v with 'i'. Since all positive hyperedges were awarded, there must be such a bidder i for each set S_v, so all vertices in G are be given a label. Further, since no negative hyperedge was awarded, and negative hyperedges were formed from pairs of items in positive hyperedge-sets S_u and S_v, it must be that no two adjacent positive hyperedge-sets were awarded to the same person. Since these positive hyperedge-sets correspond exactly to the vertices in G, no two adjacent vertices in G are given the same label in our labelling. The labelling is a 3-coloring of G, so $\langle G \rangle \in$ 3-Colorability. □

6 Breadth-Restricted Overlap

In this section, we give a polynomial-time algorithm for the restriction $\Delta(G_h) \leq 2$.

6.1 Algorithm

We consider connected components of the graph G_h, ordered by the number of items in the union of the hyperedges in the corresponding hypergraph representation h. Let $C_1, C_2, \ldots, C_K$ be the resulting ordering. This way, for some connected component C_k, $k \leq K$, any connected component contained inside one of the regions of C_k will occur earlier in the ordering. If the set G has a nonzero hyperedge weight, then C_K corresponds to the hyperedge G. Otherwise, we treat it as a hyperedge nonetheless, corresponding to the "dummy" connected component C_G, and we append C_G to the list.

We find that we can directly apply some of the recurrences from section 4 by inserting a "dummy" (zero-valued) hyperedge around each connected component C_k. We reuse the following

recurrences:

$$\mathcal{A}_{\text{OPT}}^{S_h} = \max(\mathcal{A}_{\text{B}}^{S_h}, \mathcal{A}_{\text{ALT}}^{S_h}).$$
$$\mathcal{A}_{\text{B}}^{S_h} = S_h \times \{i^{(1)}(S_h)\}.$$
$$\mathcal{A}_{\text{B}'}^{S_h} = S_h \times \{i^{(2)}(S_h)\}.$$

However, we rewrite $\mathcal{A}_{\text{ALT}}^{S_h}$ as follows:

$$\mathcal{A}_{\text{ALT}}^{S_h} = \max(\mathcal{A}_{\text{B}'}^{S_h}, \mathcal{A}_{\text{NB}}^{S_h}),$$

where $\mathcal{A}_{\text{NB}}^{S_h}$ gives the optimal allocation in which S_h is allocated to more than one bidder.

To solve $\mathcal{A}_{\text{NB}}^{S_h}$, we revert back to viewing h as a connected component C_k. Observe that by the constraint that $\Delta(G_h) \leq 2$, one of the following must be true:

1. C_k consists of hyperedges $h_1, h_2, \ldots, h_\gamma$ such that the only intersections are between pairs h_α, $h_{\alpha+1}$, $\alpha < \gamma$.
2. C_k consists of hyperedges $h_1, h_2, \ldots, h_\gamma$ such that the only intersections are between pairs h_α, $h_{\alpha+1}$, $\alpha < \gamma$, and between h_1 and h_γ.
3. C_k consists of three hyperedges intersection in Venn diagram-style.

We present a recurrence relation $\mathcal{A}_{\text{NB}}^{S_h}$ for case 1, then show how we can call the recurrence polynomially many times to obtain solutions for cases 2 and 3.

Where h is associated with connected component C_k, we first give $\mathcal{A}_{\text{OPT}}^{S_h}$ then later show how we can augment the algorithm to find $\mathcal{A}_{\text{NB}}^{S_h}$.

Let C_k consist of hyperedges $h_1, \ldots, h_\gamma$. Then,

$$\mathcal{A}_{\text{OPT}}^{S_h} = \mathcal{A}_{\text{OPT}}^{S_\gamma},$$

where we define

$$S_\alpha = \bigcup_{a=1}^{\alpha} S_{h_\alpha}.$$

Then $\mathcal{A}_{\text{OPT}}^{S_\alpha}$ is given by

$$\mathcal{A}_{\text{OPT}}^{S_\alpha} = \max\big(\max_i \mathcal{A}_{\text{OPT,i}}^{S_{\alpha-1}} \cup [S_{h_\alpha} \setminus S_{h_{\alpha-1}} \times \{i\}], \tag{2}$$
$$\mathcal{A}_{\text{OPT}}^{S_{\alpha-1}} \cup \mathcal{A}_{\text{G}}^{S_{h_\alpha} \setminus S_{h_{\alpha-1}}}, \tag{3}$$
$$\mathcal{A}_{\text{OPT}'}^{S_{\alpha-1}} \cup \mathcal{A}_{\text{G}}^{S_{h_\alpha} \setminus S_{h_{\alpha-1}}}, \tag{4}$$
$$\mathcal{A}_{\text{OPT}}^{S_{\alpha-1}} \cup \mathcal{A}_{\text{GS}}^{S_{h_\alpha} \setminus S_{h_{\alpha-1}}}\big) \tag{5}$$

where $\mathcal{A}_{\text{OPT,i}}^{S_\alpha}$ is the optimal allocation of S_α conditioned on the goods in $S_{h_\alpha} \cap S_{h_{\alpha+1}}$ being allocated to bidder i, and $\mathcal{A}_{\text{OPT}'}^{S_{\alpha-1}}$ is the optimal allocation of S_α conditioned on the goods in $S_{h_\alpha} \cap S_{h_{\alpha+1}}$

being allocated *differently* than in $\mathcal{A}^{S_\alpha}_{\text{OPT}}$.

$\mathcal{A}^{S_\alpha}_{\text{OPT,i}}$ simply forces a region of items to be allocated to bidder i. Therefore, we can compute this allocation by running $\mathcal{A}^{S_\alpha}_{\text{OPT}}$ on a version of the hypergraph in which the items in $S_{h_\alpha} \cap S_{h_{\alpha-1}}$ are removed. Combine the resulting allocation with $(S_{h_\alpha} \cap S_{h_{\alpha-1}}) \times \{i\}$ to obtain the desired allocation.

Now that we are able to compute the welfare maximizing allocation for case 1, we describe how to augment the algorithm to find the welfare maximizing allocation for cases 2 and 3.

Case 2. Case 2 is identical to case 1 except that h_1 intersects with h_γ—call that intersecting region R. While there are more efficient options, an easy polynomial-time solution to this problem is to run the algorithm for case 1 $n+1$ times: for each bidder i, run the algorithm conditioned on R being fully allocated to bidder i, and then run the algorithm conditioned in R being allocated optimally to at least two bidders (this may require computing a minimum-cost swap). Of the $n+1$ tentative allocations, select the one which maximizes welfare.

Case 3. Case 3 is unique in that there is a region—call it R'—that is the intersection of three hyperedges. However, observe that without this region, the hypergraph would fall into case 2. Therefore, we run the algorithm described in case 2 $n+1$ times, each time conditioning on R' being allocated to a particular bidder, or to be split among multiple bidders.

Finally, we argue how to transform the recurrence for $\mathcal{A}^{S_h}_{\text{OPT}}$ to a recurrence for $\mathcal{A}^{S_h}_{\text{NB}}$. The two allocations only differ if $\mathcal{A}^{S_h}_{\text{OPT}}$ allocates all items in the connected component to a single bidder. To guard against this, we assert the following for each allocation $\mathcal{A}^{S_\alpha}_{\text{OPT}}$, $\mathcal{A}^{S_\alpha}_{\text{OPT,i}}$, and $\mathcal{A}^{S_\alpha}_{\text{OPT}'}$ at every step α:

If the allocation gives all items to a single bidder, compute the second-best allocation as well.

At each step, this will require (1) computing the second-best bundle allocation and (2) computing a swap. Since we compute these values for every α, we dynamically store whether the swap should occur in $S_{h_\alpha} \setminus S_{h_{\alpha-1}}$ or $S_{h_\alpha} \cap S_{h_{\alpha-1}}$, exactly as in (4) and (5).

6.2 Correctness

We show that $\mathcal{A}^{S_\alpha}_{\text{OPT}}$ correctly computes the assuming correct values for $\mathcal{A}^{S_{\alpha-1}}_{\text{OPT,i}}$ and $\mathcal{A}^{S_{\alpha-1}}_{\text{OPT}'}$.

In the optimal allocation, the goods in S_α fall into one of the following cases:

1. S_α is allocated in full to some bidder i.
2. S_α is allocated greedily based on the allocations computed in C_{k-1}.
3. S_α is allocated greedily based on the allocations computed in C_{k-1}, but one of those allocations needed to be switched in order to avoid awarding a negative hyperedge. This swap occurred

in $S_{h_\alpha} \cap S_{h_{\alpha-1}}$.

4. S_α is allocated greedily based on the allocations computed in C_{k-1}, but one of those allocations needed to be switched in order to avoid awarding a negative hyperedge. This swap occurred in $S_{h_\alpha} \setminus S_{h_{\alpha-1}}$.

We find that the cases above line up well with the expressions in (2)-(5).

In case 1, the optimal allocation is found by awarding all goods in S_{h_α} to the same bidder. This will require the items in $S_{h_\alpha} \cap S_{h_{\alpha-1}}$ to be awarded to some bidder i. Since $\mathcal{A}^{S_{\alpha-1}}_{\mathrm{OPT,i}}$ computes the optimal allocation under this condition, $\max_i \mathcal{A}^{S_{\alpha-1}}_{\mathrm{OPT,i}}$ finds the optimal bundle allocation.

Case 2 places no restrictions on the items in $S_{\alpha-1}$. Therefore, the optimal allocation is found by combining the optimal allocation of $S_{\alpha-1}$ with the greedy allocation of $S_{h_\alpha} \setminus S_{h_{\alpha-1}}$.

Case 3 enforces that the region $S_{h_\alpha} \cap S_{h_{\alpha-1}}$ must not entirely be allocated to the same bidder that the rest of S_α is, but otherwise places no restrictions on the allocation. Therefore, the optimal allocation is found by combining $\mathcal{A}^{S_{\alpha-1}}_{\mathrm{OPT}'}$ with the greedy allocation for the remaining items.

Similarly, case 4 enforces that the region $S_{h_\alpha} \setminus S_{h_{\alpha-1}}$ must not entirely allocated to the same bidder that $S_{h_\alpha} \cap S_{h_{\alpha-1}}$ is. Thus, the optimal allocation is found by allocating S_α optimally and swapping some item in $S_{h_\alpha} \setminus S_{h_{\alpha-1}}$.

References

[1] Abraham, I., Babaioff, M., Dughmi, S., & Roughgarden, T. (2012, June). Combinatorial auctions with restricted complements. In Proceedings of the 13th ACM Conference on Electronic Commerce (pp. 3-16). ACM.

[2] Blumrosen, L., & Nisan, N. (2007). Combinatorial auctions. Algorithmic game theory, 267, 300.

[3] Boutilier, C., & Hoos, H. H. (2001, August). Bidding languages for combinatorial auctions. In International Joint Conference on Artificial Intelligence (Vol. 17, No. 1, pp. 1211-1217). LAWRENCE ERLBAUM ASSOCIATES LTD.

[4] Cavallo, R., Parkes, D. C., Juda, A. I., Kirsch, A., Kulesza, A., Lahaie, S., ... & Shneidman, J. (2005). TBBL: A tree-based bidding language for iterative combinatorial exchanges.

[5] Conitzer, V., Sandholm, T., & Santi, P. (2005, July). Combinatorial auctions with k-wise dependent valuations. In AAAI (Vol. 5, pp. 248-254).

[6] Dobzinski, S., Nisan, N., & Schapira, M. (2005, May). Approximation algorithms for combinatorial auctions with complement-free bidders. In Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (pp. 610-618). ACM.

[7] Feige, U., Feldman, M., Immorlica, N., Izsak, R., Lucier, B., & Syrgkanis, V. (2014). A unifying hierarchy of valuations with complements and substitutes. arXiv preprint arXiv:1408.1211.

[8] Lehmann, B., Lehmann, D., & Nisan, N. (2001, October). Combinatorial auctions with decreasing marginal utilities. In Proceedings of the 3rd ACM conference on Electronic Commerce (pp. 18-28). ACM.]

[9] Nisan, N. (2000, October). Bidding and allocation in combinatorial auctions. In Proceedings of the 2nd ACM conference on Electronic commerce (pp. 1-12). ACM.

[10] Nisan, N. (2006). Bidding languages. Combinatorial auctions. Cambridge: MIT.

[11] Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. Artificial intelligence, 135(1-2), 1-54.

A Expressivity of the Hypergraph Representation

In this section, we explore the expressivity of the hypergraph representation by examining well-known valuation classes and their corresponding restrictions on the hypergraph.

A.1 Additive Valuations

Additive valuations are described very simply by hypergraphs with no hyperedges. The value of a set S is given by

$$v(S) = \sum_{j \in S} v(j),$$

the sum of the values on each of the vertices in S. Therefore, hypergraphs can represent additive valuations in linear space and compute the value of any set in linear time.

A.2 Unit Demand

While hypergraphs very neatly represent additive valuations, in general they require exponential storage to represent the *unit demand* valuation. This is a scenario in which the Maximum over Positive Hypergraph representation is more concise, only requiring linear storage. The *unit demand* valuation class is defined as follows:

$$v(S) = \max_{j \in S} v_i(j)$$

for any $S \subseteq G$, $j \in G$. Observe that for a set of two items, $S = \{j, j'\}$, the following equality must hold:

$$w(\{j\}) + w(\{j'\}) + w(\{j, j'\}) = \max(v_i(j), v_i(j')).$$

Since $w(\{j\}) = v_i(j)$ and $w(\{j'\}) = v_i(j')$, this gives the following value for $w(\{j, j'\})$:

$$w(\{j, j'\}) = -\min(v_i(j), v_i(j')).$$

Claim 1. *For a set S of size m,*

$$w(S) = (-1)^{m+1} \min_{j \in S}(v_i(j)).$$

Proof. Consider a set S and let j^* be its highest-valued item,

$$v_i(S) = v_i(j^*).$$

Using the definition of hyperedges and unit demand, we have that

$$\begin{aligned} v_i(S) = v_i(j^*) &= \sum_{S' \subseteq S \setminus \{j^*\}} [w(S') + w(S' \cup \{j^*\})] \\ &= w(j^*) + w(S \setminus \{j^*\}) + w(S). \end{aligned}$$

The majority of the terms cancel because the only difference between $w(S')$ and $w(S' \cup j^*)$ is the difference in sign (because j^* is the highest-valued good and so cannot be the minimum). The only cases where this isn't true is for $S' = \varnothing$, which leaves us with a lone $w(j^*)$ term, and $S' = S$, which we cannot use the inductive hypothesis for, so we are left with $w(S \setminus \{j^*\}) + w(S)$. Since the terms $v_i(j^*)$ and $w(j^*)$ cancel, we are left with

$$w(S) = -w(S \setminus \{j^*\}),$$

where j^* is guaranteed not to be the minimum element in S. This proves the result. □

A.3 Subadditive and superadditive

We think of positive and negative hyperedges as representing superadditive and subadditive valuations, respectively. If we restrict ourselves to two goods, say A and B, then the valuation is superadditive if $w(\{A, B\}) \geq 0$ and subadditive if $w(\{A, B\}) \leq 0$. Thus it is natural to ask the question of whether *positive hypergraphs* (hypergraphs with nonnegative edge weights) can only represent superadditive valuations, and whether *negative hypergraphs* (hypergraphs with nonpositive edge weights, except degree 1 hyperedges) can only represent subadditive valuations. We answer this question with the following examples.

From the 2 good examples it would appear that negative hypergraphs correspond to subs *Subadditivity.* We argue that subadditive hypergraph valuations are classified by the following:

$$\sum_{S' \subseteq S \cap T} w(S') + \sum_{S' \subseteq S \Delta T : S' \cap (S \setminus T) \neq \varnothing, S' \cap (T \setminus S) \neq \varnothing} w(S')$$

for all $S, T \subseteq G$.

Proof. By definition, subadditive valuations are described by the following inequality:

$$v(S) + v(T) \geq v(S \cup T)$$

for all $S, T \subseteq G$. In hyperedge terms, this becomes

$$\sum_{S' \subseteq S} w(S') + \sum_{S' \subseteq T} w(S') \geq \sum_{S' \subseteq S \cup T} w(S').$$

The LHS double-counts all valuations of subsets in $S \cap T$. However, the RHS includes *some* valuations of subsets in $S \Delta T$—specifically those with some element in $S \setminus T$ *and* some element in $T \setminus S$. Subtracting all common terms from both sides gives the desired inequality

$$\sum_{S' \subseteq S \cap T} w(S') \geq \sum_{S' \subseteq S \Delta T : S' \cap (S \setminus T) \neq \varnothing, S' \cap (T \setminus S) \neq \varnothing} w(S').$$

□

Superadditivity. By symmetry with the subadditive case, superadditive hypergraph valuations are classified by the following:

$$\sum_{S' \subseteq S \cap T} w(S') \leq \sum_{S' \subseteq S \Delta T : S' \cap (S \setminus T) \neq \varnothing, S' \cap (T \setminus S) \neq \varnothing} w(S')$$

for all $S, T \subseteq G$.

While in the case of two goods, the definitions of subadditivity and superadditivity coincide with the restrictions of negative and positive hypergraph valuations, respectively. Sadly this observation does not generalize.

Claim 2. *Positive hyperedge weights $\not\Rightarrow$ superadditivity.*

Proof. Consider the example of three goods a, b and c, with $v(a) = v(c) = 1$ and $v(b) = 100$. Let $w(\{a, b\}) = w(\{b, c\}) = w(\{a, c\}) = w(\{a, b, c\}) = 0$. Then

$$v(\{a, b\}) + v(\{b, c\}) = 202 > 102 = v(\{a, b, c\}).$$

This violates the superadditivity condition. □

Claim 3. *Negative hyperedge weights $\not\Rightarrow$ subadditivity.*

Proof. Consider four goods, a, b, c, and d with $v(a) = v(b) = v(c) = v(d) = 1$ and with a single hyperedge $w(\{b, d\}) = -100$. Then

$$v(\{a, b, d\}) + v(\{b, c, d\}) = -194 < -96 = v(\{a, b, c, d\}).$$

This violates the subadditivity condition. □

However, this example does not seem intuitively sensible because someone would end up with -98 utility for items b and d.

A.4 Monotone

We conclude this section with the most general valuation class we study: the monotone class of valuations. Monotone valuations are characterized by the following:

$$v(T_1) \leq v(T_2) \text{ whenever } T_1 \subseteq T_2.$$

We will show that for a hypergraph representation w, this is equivalent to the following:

$$\sum_{S' \subseteq S} w(S \cup \{j\}) \geq 0$$

for any set $S \subseteq G$ and item $j \in G$.

Lemma A.1. *A valuation function v is monotone if and only if*

$$v(S) \leq v(S \cup \{j\}) \text{ for any } j \in G,\ S \subseteq G.$$

Proof of Lemma. The traditional definition of monotonicity is that

$$v(T_1) \leq v(T_2) \text{ whenever } T_1 \subseteq T_2.$$

We show that if $v(T_1) > v(T_2)$ for $T_1 \subset T_2$, then the condition

$$v(S) \leq v(S \cup \{j\}) \text{ for any } j \in G,\ S \subset G$$

is violated for some set S and item j.

Let $T \subsetneq S$ and $S \setminus T = \{x_1, x_2, \ldots, x_N\}$. Consider the sequence

$$v(T_1), v(T_1 \cup \{x_1\}), v(T_1 \cup \{x_1, x_2\}), \ldots, v(T_2).$$

If $v(T_1) > v(T_2)$, then there must be some K such that $v(T \cup \{x_1, \ldots, x_{K-1}\}) > v(T \cup \{x_1, \ldots, x_K\})$. Letting $S = \{x_1, \ldots, x_{K-1}\}$ and $j = x_K$, this completes the proof. □

Claim 4. *For a valuation function v with hypergraph representation w, v is monotone if and only if*

$$\sum_{S' \subseteq S} w(S \cup \{j\}) \geq 0$$

for any set $S \subseteq G$ and item $j \in G$.

Proof. From the above lemma, we have that

$$v(S) \leq v(S \cup \{j\}) \text{ for any } j \in G,\ S \subset G.$$

Written in terms of hyperedges, this is

$$\sum_{S' \subseteq S} w(S') \leq \sum_{S' \subseteq S \cup \{j\}} w(S')$$

or

$$\sum_{S' \subseteq S} w(S \cup \{j\}) \geq 0.$$

□

In the previous section, restricting the hyperedges to positive or negative values failed to restrict the class of valuations. However, Claim 4 gives the following corollary, which adds some characterization to negative hypergraphs.

Corollary A.2. *Valuation functions described by positive (nonnegative) hypergraphs are monotone.*