Computational Bundling for Auctions

Christian Kroer **Tuomas Sandholm**

May 7, 2013
CMU-CS-13-111

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work supported by the National Science Foundation under grants CCF-1101668 and IIS-0964579. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Keywords: Bundling, search, integer programming, combinatorial auctions, VCG, revenue maximization

Abstract

Revenue maximization in combinatorial auctions (and other multidimensional selling settings) is one of the most important and most elusive problems in mechanism design. The design problem is NP-complete, and the optimal designs include features that are not acceptable in many applications, such as favoring some bidders over others and randomization. In this paper, we instead study a common revenue-enhancement approach - bundling - in the context of the most commonly studied combinatorial auction mechanism, the Vickrey-Clarke-Groves (VCG) mechanism. A second challenge in mechanism design for combinatorial auctions is that the prior distribution on each bidder's valuation can be doubly exponential. Such priors do not exist in most applications. Rather, in many applications (such as premium display advertising markets), there is essentially a point prior, which may not be accurate. We adopt the point prior model, and prove robustness to inaccuracy in the prior. Then, we present a branch-and-bound framework for finding the optimal bundling. We introduce several techniques for branching, upper bounding, lower bounding, and lazy bounding. Experiments on CATS distributions validate the approach and show that our techniques dramatically improve scalability over a leading general-purpose MIP solver.

1 Introduction

Revenue maximization in combinatorial auctions (and other multidimensional selling settings) is one of the most important and most elusive problems in mechanism design. The optimal auction for a single item is known [41] and has been generalized to multiple units of one item [38], but the problem remains open even with just two items. The fact that the general optimal combinatorial auction mechanism is unknown is not a coincidence: even a special case of that design problem is NP-complete, even in the private values setting [18]. This suggests that, unlike for single-item settings, a concise characterization of optimal combinatorial auctions cannot exist (unless P=NP). This is one of the key motivations for *automated mechanism design* where an algorithm is used to design the mechanism for the setting (prior probability distribution) at hand (e.g., [16, 17, 18, 50, 35, 36, 13]).

Even with automated design, and putting aside the computational complexity of the design problem, it is not clear that optimal combinatorial auctions are viable in practice, for the following reasons.

1. The revenue-optimal mechanism includes features that are not acceptable in many applications, such as favoring some bidders over others, and randomization.

2. The optimal mechanism is difficult to understand. This, itself, can be a deterrent to its adoption.

3. Even in the private values setting, the prior distribution on each bidder's valuation can have support the size of which is doubly exponential. Specifically, if there are m items and a bidder can have any of k values for each bundle, the support of the prior has k^{2^m-1} points because that is the size of the bidder's type space. Such prior distributions have not been (and cannot be) constructed in most applications.

In this paper, we study a practical automated mechanism design setting where we avert these problems.

We avert the first two problems by only considering one common, practical way of increasing revenue, *bundling*. Our mechanisms will be fair in the sense that they are symmetric across bidders, and deterministic, unlike the optimal auction. Specifically, we will develop algorithms for optimal bundling in the context of the most commonly studied combinatorial auction mechanism, the *Vickrey-Clarke-Groves mechanism (VCG)* [58, 15, 24].[1] In the VCG—even with bundling—each bidder's dominant strategy is to bid truthfully.

We avert the third problem by not assuming that we have access to such a prior. In many (arguably most) applications there is essentially just a point prior, and it may not be accurate. In other words, the seller has expectations about how much bidders would be willing to pay for various bundles, but the seller does not have a sophisticated probabilistic model about any bidder's valuations (which are not independent across bundles). This is the case, for example, in TV advertising sales and in premium, guaranteed display advertising sales. (In both of those markets, the sales

[1] While the VCG itself has taken some time to get adopted in applications, it is has been successfully adopted, for example, by Facebook for its advertising auctions.

occur manually, and the inventory is implicitly bundled in ad hoc ways today, with typically 2-4 targeting attributes.[2]) Therefore, we adopt the point prior model. However, we also acknowledge the fact that in practice the point prior might not be accurate; we prove robustness of our approach to inaccuracy in the prior. One might also ask why we do not simply make take-it-or-leave-it offers to take the entire surplus for the seller given that we have a point prior. The reason we do not use that mechanism is that it is highly nonrobust to error in the prior: even slight overpricing on a bundle will cause the revenue from that bundle to drop to zero.

We develop a custom branch-and-bound framework for finding the optimal bundling. We show that algorithms in that framework scale significantly better than a leading general-purpose integer program solver, CPLEX. We design and compare several techniques for upper bounding, lower bounding, branching, and lazy child node evaluation. Experiments on the leading combinatorial auction test suite, CATS [34], validate the approach.

Our approach to bundling is computational. The goal is not to build insight from manual analysis that can then loosely be applied to practice—although the computational approach can help develop insight as well and fuel future theory. Rather, the goal is to develop a computational methodology that can be used in practice. It is therefore key that we make the computational techniques as scalable as possible.

1.1 Prior research on bundling

Most prior work on bundling has been analytical rather than computational, and has therefore mainly studied simpler bundling settings than ours.

1.1.1 Bundle pricing (in posted prices)

Most prior work on bundling has been in the context of posted prices, that is, take-it-or-leave-it offers, as in, for example, catalog pricing.

The first mention of being able to increase revenue via bundling is attributed to economist George J. Stigler in his 1963 discussion of anti-trust Supreme Court rulings (the issue was whether a movie studio should be allowed to bundle *Gone with the Wind* with another movie) [55]. Bundle pricing in economics has often focused on analyzing two-item settings to provide insight into the way monopolies can improve profits by offering goods in bundles [1, 19, 25, 39, 54]. (One exception is that Armstrong [2] examines m-item settings, but places severe restrictions on buyers' utility functions. Another exception is that Manelli and Vincent [37] provide results for when bundled catalog sales are optimal, mainly in the two- and three-item settings.) This work provides sufficient conditions on when bundling is profitable and optimal pricing strategies under various assumptions.

There has been some computational work on bundle pricing. For example, Hanson and Martin [27] present a mixed integer program for optimizing bundle prices for a handful of market seg-

[2]This is in stark contrast to display ad exchanges where campaigns with guarantees cannot be bought. Rather, each impression is sold on the fly as users visit web sites. The sale is typically conducted using a second-price auction. The bidders are allowed to use larger numbers of targeting attributes to determine their bids [20, 40]. Premium display advertising content is typically sold in the manual market and the exchanges get mainly remnant inventory.

ments. They assume that each of the segments can be described by a single value for each bundle, and that the value of every bundle for every market segment is known in advance. They also do not describe how their bundle pricing strategy compares to using item prices. Rusmevichientong et al. [46] investigate the problem of pricing different car configurations based on customer survey data collected by GM's Auto Choice Advisor web site. An extensive revenue management literature also exists, but it tends to focus on pricing individual items in the face of stochastic demand and limited supply, even when there are multiple distinguishable items for sale [56, 22]. Two notable exceptions consider the problem of selling two products under different bundling policies [12, 57]. There has also been work on deriving valuation models from data, and on using those models to price items and bundles. For example, Jedidi et al. [30] fit a customer valuation model for two items; the need for non-trivial amounts of survey data prevent that method from being fully automated and scalable. Benisch and Sandholm [9] present a framework for automatically suggesting high-profit bundle discounts based on historical customer purchase (shopping cart) data. They develop techniques for fitting a probabilistic valuation model to the data, and search algorithms that identify profit-maximizing prices and bundle discounts given the model. Computational experiments demonstrate conditions under which offering discounts on bundles can benefit the seller, the buyer, and the economy as a whole. In contrast to products typically suggested by recommender systems, the most profitable products to offer bundle discounts on appear to be those that are occasionally purchased together and often separately.

There has also been work on pricing bundles of information goods specifically, where it is usually assumed that customers care only about how many goods are bundled together (i.e., their valuation for a bundle depends only on its size, not its contents) and there are no marginal costs. For example, Kephart et al. [33] and Brooks and Durfee [11] describe online approaches to pricing in this domain. Additionally, Bakos and Brynjolfsson [6] provide an analytical treatment of this problem with some valuable insights about when bundling is profitable. Hitt and Chen [28] and Wu et al. [60] consider a bundle pricing mechanism for information goods that allows customers to choose up to m' items from a larger pool of m items.

Related pricing work in theory of computer science has focused primarily on pricing items rather than bundles, and for single-minded customers that desire only one bundle. For example, Balcan and Blum [7] and Balcan et al. [8] provide online and approximate algorithms for this setting, and Guruswami et al. [26] show that finding the optimal pricing is APX-hard.

Walsh et al. [59] study inventory bundling in premium display advertisement campaign selling, with the goal of making the winner determination problem smaller and easier. That goal is different than revenue maximization.

1.1.2 Bundling in auctions

There has been some work on bundling in auctions. Palfrey [43] shows that bundling all items is better than selling them separately if there are only two bidders. Chakraborty [14] shows that under certain assumptions there is a critical number of bidders above which separate sales yield higher revenue than pure bundling, and vice versa. Armstrong [3] shows that in a two-item auction with two possible valuations per bidder, the revenue-maximizing auction is efficient. Avery and Hendershott [5] show that even in a two-item setting, this is not the case in general, and that unlike

in posted pricing where optimal bundling decisions are deterministic [39], in auctions, randomized bundling can increase revenue. (A buyer may then receive a discount on a lower-valued product without receiving a higher-valued product.) Automated mechanism design also produced randomized mechanisms for revenue-maximizing combinatorial auctions [17]. Since our paper focuses on the VCG mechanism, our bundling is deterministic.

There has also been work on generalizing the VCG to higher-revenue dominant-strategy auction mechanisms (which do bundling as a side effect of running the mechanism). Likhodedov and Sandholm [35, 36] study a generalization of VCG called *virtual valuations combinatorial auctions (VVCAs)* and a generalization of them, *affine maximizer combinatorial auctions*, which are auctions where the VCG is run on affine transformations of the bidders' bids. That work includes analysis and automated auction design algorithms. Those mechanism design algorithms scaled to a handful of items. For the restricted setting of additive valuations, Jehiel et al. [31] analyze a subclass of VVCAs called λ-*auctions*, which are nevertheless rich enough to allow the bundling to depend on the bids. Our approach of bundling first and then simply running VCG is easier to for buyers and sellers to understand. Also, none of the prior papers present any bundling algorithms.

Ghosh et al. [23] study bundling in sponsored search auctions with contexts. They prove hardness results and design approximation algorithms.

To our knowledge, there has been no prior work in the setting of our paper.

2 Notation and problem formulation

We assume that we have a set of bidders $N = \{1, \ldots, n\}$, a set of items $M = \{1, \ldots, m\}$, and a set of bids B. (The bids represent the point prior.) B_i is the set of all bids for bidder i, and B_{-i} the set of all bids that do not belong to i. A bid is a tuple $\langle S_j, v_j \rangle$, where $S_j \subseteq M$ is the set of items that the bid wants and $v_j \geq 0$ is a valuation.

We denote an allocation of items to bidders by α, and α_i is the set of items allocated to bidder i in the allocation. We overload $v_i(\alpha)$ to be bidder i's valuation for α_i.

We work in the standard combinatorial auction setting with the XOR bidding language [49] so each bidder can have at most one of her bids win. Every item can be assigned to at most one bidder. There are *no externalities*: the valuation for each bidder i depends only on the items that i receives. There is *free disposal*: the value of a subset of items $M' \subseteq M$ for bidder i is less than (or equal to) the value of M.

A bundle $b \subseteq M$ is a set of items from M. A *bundling* ϕ is a set of bundles that partitions M. We denote the set of all possible bundlings by Φ.

We now define two different notions of overlap in bids, which we will use throughout the paper.

Definition 1 (intersect)**.** *Two bids i, j* intersect *if their item sets overlap: $S_i \cap S_j \neq \emptyset$.*

Bundling can, in effect, introduce additional overlap between bids, and for this reason we need a more general notion of overlap:

Definition 2 (bundling-intersect)**.** *Two bids i, j bundling-intersect in a given bundling ϕ if they intersect or there exists at least one item in each of the two bids so that those two items have*

been included in the same bundle. Formally, two bids i, j bundling-intersect in bundling ϕ if they intersect or $\exists a_i \in S_i, a_j \in S_j, b \in \phi$ such that $a_i \in b, a_j \in b$. We denote the set of bundling-intersecting items of the two sets by $S_i \sqcap S_j$.

We will be running the VCG to auction the bundles simultaneously. In the vanilla VCG without bundling, the allocation of items to bidders is computed so that it maximizes social welfare (henceforth referred to simply as welfare): $\alpha^* = \max_\alpha \sum_{i=1}^n v_i(\alpha)$. The payment from each bidder i is $p_i = W^{*,-i}_{M,B} - W^{*,-i}_{M_{-i},B}$, where $W^{*,-i}_{M,B}$ is the optimal welfare where i receives no items, and $W^{*,-i}_{M_{-i},B} = \sum_{j \neq i} v_j(\alpha^*)$ is the welfare of the other agents in α^*. We denote by $W_B(\alpha)$ the welfare of an allocation α for bids B.

In our setting, we have to take the bundling into account in the VCG. For a given bundling $\phi \in \Phi$, the VCG allocation α^* is computed as before, but with the added constraint that no two items that are in the same bundle can be allocated to different bidders (i.e., for any two winning bids i and j, $S_i \sqcap S_j = \emptyset$). We denote the welfare of such an allocation by $W^*_{\phi,B} = \max_{\alpha \in A_\phi} \sum_{i \in N} v_i(\alpha)$, and the welfare of the welfare-maximizing allocation over the bundling, where bidder i is excluded, as $W^{*,-i}_{\phi,B} = \max_{\alpha \in A_\phi} \sum_{j \neq i} v_j(\alpha)$. Similarly, running VCG on bundling ϕ yields payments $p_i = W^{*,-i}_{\phi,B} - W^{*,-i}_{\phi_{-i},B}$ from each bidder i. Here, ϕ_{-i} is the set of bundles from ϕ that are not allocated to bidder i. The goal in optimal bundling is to find a bundling ϕ^* such that the revenue $r_{\phi^*} = \sum_{i=1}^n p_i$ is maximized.

3 Basic properties of the approach

In this section we study some important basic properties of the proposed approach.

3.1 Number of bundlings

The number of bundlings grows extremely rapidly as the number of items grows. The number of ways to bundle (that is, exhaustively partition) m items is called the Bell number, BELL_m, of m. The Bell numbers can be defined recursively: $\text{BELL}_m = \sum_{k=0}^{m-1} \binom{m-1}{k} \text{BELL}_k$. Sandholm et al. [52] proved that $\text{BELL}_m \in \omega(m^{\frac{m}{2}})$. Berend and Tassa [10] proved that $\text{BELL}_m < (\frac{0.792m}{\ln(m+1)})^m$.

3.2 NP-hardness

We first prove that this bundling problem (i.e., revenue-maximizing bundling in the VCG setting with a point prior) is NP-hard. This is the case even if one could compute the revenue of a given bundling in polynomial time.

Theorem 3.1. *Finding the optimal bundling is NP-hard, regardless of the hardness of computing the revenue of a given bundling.*

Proof. The proof is by reduction from bin packing. The bin packing problem is the following: Given a set of n objects of sizes $x_1, ..., x_n$, and positive integers k and V, is it possible to fit the n objects into k bins of size V?

For each bin $i = 1, ..., k$, we generate $n + 1$ items: a_i^{bin} and $G_i = \{g_1^i, ..., g_n^i\}$, and two bidders with bids $b_i^{bin}, b_{i'}^{bin}$ respectively, where $S_i^{bin} = \{a_i^{bin}\} \cup G_i, S_{i'}^{bin} = \{a_i^{bin}\}$, and valuations $v_i^{bin} = V + M, v_{i'}^{bin} = M$, for some $M > V, M > \sum_{j=1}^n x_j$. For each of the objects $j = 1, ..., n$, we generate an item a_j^{obj} and a bidder with bid b_j^{obj} with $S_j^{obj} = \{a_j^{obj}\}$, and valuation $v_j^{obj} = x_j$. Note that the only pairs of bids that overlap before bundling are $b_i^{bin}, b_{i'}^{bin}$.

We claim that there is a solution to the bin packing problem iff there is some bundling with revenue $r = kM + \sum_{j=1}^n x_j$ for the above bundling problem. Now, assume that we have some solution with revenue $r \geq kM + \sum_{j=1}^n x_j$. Clearly, it cannot be the case that any two items such that $S_i^{bin} \cap S_j^{bin} \neq \emptyset$ for some bids b_i^{bin}, b_j^{bin} are bundled together, as this would cause revenue $r \leq (k-1)M + \sum_{j=1}^n x_j < kM$, since only $k - 1$ bids with valuation above or equal to M can now win, and the remaining bids can have aggregate valuation at most $\sum_{j=1}^n x_j < M$. Hence, we are guaranteed kM revenue from the bids $\{b_i^{bin}, b_{i'}^{bin} : i = 1, ..., k\}$, and we know that they will not be bundled together. Now, each of the bids b_j^{obj} must be contributing x_j to revenue (either by paying that amount or causing one other bidder to pay that much more), since they are the only bids left that can be made bundling-intersecting with any other bid through bundling, and they have no competition if unbundled. We also know that any two bids b_i^{obj}, b_j^{obj} cannot be made bundling-intersecting. If they were, the revenue obtained from the two would be at most $\max(v_i^{obj}, v_j^{obj})$ (since they can be made bundling-intersecting with at most one winning bid, and alternatively if one of the two bids wins, its payment cannot exceed its valuation). It follows that the bids b_j^{obj} must bundling-intersect with the bids b_i^{bin}, which are already contributing kM revenue. Let B_i^{obj} be the set of bids b_j^{obj} that bundling-intersect with b_i^{bin}. We must have $M + \sum_{j \in B_i^{obj}} x_j \leq M + V$, since otherwise $r < kM + \sum_{j=1}^n x_j$, as some x_j is then not contributing its full valuation. Now, we can take the solution and turn it into a solution to the bin packing problem. Since we know that all b_j^{obj} bundling-intersect with one b_i^{bin} each, we take each such j and assign that object to bin i. Since $\sum_{j \in B_i^{obj}} x_j \leq V$ for all i, we know that this is a valid packing.

Conversely, if there is a solution to the bin packing problem, there is a bundling such that all bids b_j^{obj} bundling-intersect with one b_i^{bin} each, and $M + \sum_{j \in B_i^{obj}} v_j^{obj} = M + \sum_{j \in B_i^{obj}} x_j \leq V + M$ for all i. Hence we can get revenue at least $kM + \sum_{j=1}^n x_j$. □

3.3 Revenue is nonmonotonic in bidders

It is known that the VCG is not revenue monotonic in bidders [4], that is, adding bidders can decrease revenue. In fact, no dominant-strategy mechanism that satisfies participation, consumer sovereignty, and weak maximality is revenue monotonic [44]. In our setting, even with an accurate point prior, our approach does not satisfy weakly maximality, so that result does not apply. Nevertheless, we can show revenue nonmonotonicity in our setting. Consider the valuations in Figure 1 Left. The optimal bundling is to sell the items separately, in which case bidders 3 and 4 receive items Y and X respectively, with payments 3 and 5, yielding total revenue of 8. If we remove bidder 4, the optimal bundling is still to sell the items separately, where bidder 1 wins both items, with a payment of 9, which is also the revenue. Thus removing bidder 4 can increase revenue.

	X	Y	XY
Bidder 1	0	0	10
Bidder 2	4	0	4
Bidder 3	0	5	5
Bidder 4	7	0	7

	X	Y	XY
Bidder 1	10	0	10
Bidder 2	5	0	5
Bidder 3	0	2	2
Bidder 4	0	1	1
Bidder 5	0	20	20

Figure 1: Left: Bidder valuations for a game with 4 bidders and 2 items. Right: Bidder valuations for a game with 5 bidders and 2 items. In both tables, the XY column denotes bidder valuations for the bundle.

3.4 Coarseness of the optimal bundling is nonmonotonic in bidders

Perhaps surprisingly, the optimal bundling can become coarser with the addition of a bidder. An example is given in Figure 1 Right. The optimal bundling without bidder 5 is to sell X and Y separately, whereas with 5 it is optimal to bundle X and Y together.

3.5 Low worst-case revenue

It is well known that the VCG can be arbitrarily far from optimal in terms of revenue. Consider two bidders, and items X and Y, where bidder 1 bids v_a for X and bidder 2 bids v_b for Y, and there are no other bids. In this case the VCG payments by each bidder equal 0. For settings such as this, optimal bundling can have arbitrarily high revenue lift over VCG since we can bundle X and Y, and thereby earn $\min(v_a, v_b)$ for example using the Vickrey auction on the bundle.

Even with optimal bundling in the VCG, we can have arbitrarily high loss compared to what is possible. This is because the seller would give away $|v_1 - v_2|$ of the surplus (and this part can be arbitrarily large) while with reserve prices of v_1, v_2 on the items, respectively, would allow the seller to capture the entire surplus.

3.6 Robustness of the proposed approach

Using optimal reserve prices to extract surplus is not robust against error in the point prior. For example, if the presumed winners actually have valuations even slightly lower than in the prior, they will reject all the take-it-or-leave-it offers and pay nothing.

In contrast, bundling for the VCG is robust to error in the (point) prior:

Theorem 3.2. *The revenue from the VCG with an optimal bundling (which may change based on bidder valuations) is Lipschitz continuous in the valuations of the bidders with Lipschitz constant* $n - 1$. *This bound is tight.*

To prove this, we first prove the following lemma.

Lemma 3.3. *The revenue from the VCG with any fixed bundling is Lipschitz continuous in the valuations of the bidders with Lipschitz constant* $n - 1$.

Proof. The revenue obtained from VCG on a bundling ϕ can be written as

$$r_\phi = -(n-1) \cdot W^*_{\phi,B} + \sum_{i=1}^{n} W^{*,-i}_{\phi,B} \tag{1}$$

where $W^*_{\phi,B}$ is the welfare of the welfare-maximizing allocation for the bids B, and $W^{*,-i}_{\phi,B}$ denotes the same when bidder i's bids are excluded.

Thus, it suffices to show that all the terms in Equation 1 are Lipschitz continuous in bidder valuations.

Assume that some bidder i changes his valuation for some bid j, and let B' be the new set of bids, where v'_j is the new valuation for bid j. We now show that the change in welfare $\Delta W = |W^*_{\phi,B} - W^*_{\phi,B'}|$ is bounded by $\Delta v_j = |v_j - v'_i|$. For any allocation α such that j is winning, we get that the change in welfare is Δv_j, whereas for any allocation α' such that j is not winning, the welfare remains the same. Hence, the previously winning allocation α can increase or decrease by at most Δv_j. Since $W_B(\alpha) \geq W_B(\alpha')$ for all α', the new winning allocation α^* satisfies $W_{B'}(\alpha^*) \leq W_B(\alpha) + \Delta v_j$. For the lower bound, we have $W_{B'}(\alpha) \geq W_B(\alpha) - \Delta v_j$, and hence α^* must satisfy $W_{B'}(\alpha^*) \geq W_B(\alpha) - \Delta v_j$.

Since all terms are welfare maximizations over different sets of bidders, this proves that r_ϕ is Lipshitz continuous in bidder valuations, with a Lipschitz constant of $n-1$ (because the $n-1$ term and n terms in the summation change in opposite directions, and the summation over n terms only has $n-1$ terms that can change, as $W^{*,-i}_{\phi,B}$ does not depend on v_j). □

With this lemma, we are now ready to prove Theorem 3.2.

Proof. There are two possible cases. In the first case, the optimal bundling does not change, and in the second case it changes. The proof of the first case is immediate from Lemma 3.3. For the second case, let ϕ_1 and ϕ_2 be the old and new optimal bundlings, respectively. By Lemma 3.3 we can bound the revenue of a bundling under the new valuation using the old valuation:

$$r_B(\phi) - (n-1) \cdot \Delta v_i \leq r_{B'}(\phi) \leq r_B(\phi) + (n-1) \cdot \Delta v_i$$

By optimality,

$$r_{B'}(\phi_2) \geq r_{B'}(\phi_1) \geq r_B(\phi_1) - (n-1) \cdot \Delta v_i$$

From the fact that ϕ_1 is optimal for the original bids (B) we know that $r_B(\phi_2) \leq r_B(\phi_1)$ and hence

$$r_B(\phi_2) + (n-1) \cdot \Delta v_i \leq r_B(\phi_1) + (n-1) \cdot \Delta v_i$$

By Lemma 3.3, the left hand side of this inequality is an upper bound for $r_{B'}(\phi_2)$. Thus we get

$$r_B(\phi_1) - (n-1) \cdot \Delta v_i \leq r_{B'}(\phi_2) \leq r_B(\phi_1) + (n-1) \cdot \Delta B_i$$

So, the new optimal revenue is bounded both above and below as shown above, and is (by the formulas above) Lipschitz continuous with Lipschitz constant $n-1$.

Finally, we show that the bound is tight. Consider the case with 2 items X,Y and 3 bidders $\{b_1, b_2, b_3\}$, where b_1 has valuation 1 for X, b_2 has valuation 1 for Y and b_3 has valuation $\frac{1}{2}$ for X and for Y. The optimal bundling yields revenue 1, but if b_3 increases his valuation for both items to 1, the revenue increases to 2, and the increase in revenue is $1 = \frac{1}{2} \cdot (n-1)$. □

4 Mixed integer program (MIP) formulation

One approach to finding the revenue-maximizing bundling is to formulate the problem as a mixed integer program (MIP), and then use a general-purpose MIP solver—such as CPLEX—to solve the formulation. (For a thorough textbook exposition of integer programming, see Nemhauser and Wolsey [42].) In this section we give such a formulation. Later in the paper we present custom tree search algorithms that scale significantly better than CPLEX, while still guaranteeing optimality.

Figure 2 shows our MIP formulation. The basic idea behind this MIP is that we have m potential bundles, and the boolean variables $\delta_{a,b}$ denote whether item a is assigned to bundle b. Based on these assignments, the VCG payments are computed.

To break symmetries, we only allow each item $a = 1, ..., m$ to be assigned to bundles $\{1, ..., a\}$. Furthermore, items with index $a > b$ can only be assigned to the bundle with index b if the item with index b is also assigned to the bundle.

Each bid j has boolean variables I_j and I_j^{-i} that denote whether the bid wins in the optimal allocation and optimal allocation excluding bidder i, respectively. Each bidder has boolean variables $\Pi_j(b)$ and $\Pi_j^{-i}(b)$ that denote whether the bidder is allocated bundle b in the respective allocations, and boolean variables $\pi_j(a, b), \pi_j^{-i}(a, b)$ that denote whether the bidder is allocated item a through bundle b in the respective allocations. Finally, each real-valued variable p_i denotes the payment that bidder i must make.

The objective function, (2), is the sum over the payment variables of the bidders. Constraint 3 sets the payment for bidder i equal to the externality she imposes on the other bidders, i.e., her VCG payment. Constraints 4-5 ensure that a bid j can only be winning if that bidder is assigned all the items in S_j for each allocation. Constraints 6-7 ensure that each bundle is assigned to only one bidder in each allocation. Constraints 8-11 ensure that a bidder can only receive item a through bundle b in each allocation if the bidder wins the bundle, and $\delta_{a,b} = 1$, i.e., the item is in the bundle. Constraints 12-13 ensure that each bidder wins only one item in each allocation. Constraints 14-15 ensure that each item is assigned to only one bundle and they break symmetries. Finally, Constraint 16 ensures that the welfare-maximizing allocation is chosen for each bundling ϕ by ensuring that if all the $\delta_{a,b}$ that are necessary to achieve ϕ are active, then the winning bids are active.

This model suffers from several limitations, that we believe will be present in any MIP formulation for solving the entire problem. First, the MIP has $\Omega(n|B|m^2)$ boolean variables, which rapidly becomes unmanageable. More importantly, Constraint 16 is required for every possible bundling, of which there are an extremely large number as mentioned in Section 3.1; furthermore, to generate each of these constraints, the welfare-maximizing allocation must be found, which is NP-hard in itself. This could potentially be alleviated by using constraint generation techniques (which generate constraints frugally on an as-needed basis, guided by the tentative optimal solution), but even this is unlikely to yield acceptable scalability, as each added constraint only cuts off solutions at that specific bundling, and nowhere else. In addition, this would require resolving the already large MIP every time a constraint is added.

$$\max \sum_{i=1}^{n} p_i \tag{2}$$

$$p_i \leq \sum_{j \in B_{-i}} v_j \cdot I_j^{-i} - \sum_{j \in B_{-i}} v_j \cdot I_j \quad \forall i \in N \tag{3}$$

$$I_j \leq \sum_{b=1}^{m} \pi_i(a,b) \qquad \forall i \in N, j \in B_i, a \in S_j \tag{4}$$

$$I_j^{-i} \leq \sum_{b=1}^{m} \pi_k^{-i}(a,b) \qquad \forall i,k \in N, j \in B_k, a \in S_j \tag{5}$$

$$\sum_{i=1}^{n} \Pi_i(b) \leq 1 \qquad \forall b = 1, ..., m \tag{6}$$

$$\sum_{k=1}^{n} \Pi_k^{-i}(b) \leq 1 \qquad \forall i \in N, b = 1, ..., m \tag{7}$$

$$\pi_i(a,b) \leq \Pi_i(b) \qquad \forall i \in N, a = 1, ..., m, b = 1, ..., m \tag{8}$$

$$\pi_k^{-i}(a,b) \leq \Pi_k^{-i}(b) \qquad \forall i,k \in N, a = 1, ..., m, b = 1, ..., m \tag{9}$$

$$\pi_i(a,b) \leq \delta_{a,b} \qquad \forall i \in N, a = 1, ..., m, b = 1, ..., m \tag{10}$$

$$\pi_k^{-i}(a,b) \leq \delta_{a,b} \qquad \forall i,k \in N, a = 1, ..., m, b = 1, ..., m \tag{11}$$

$$\sum_{j \in B_i} I_j \leq 1 \qquad \forall i \in N \tag{12}$$

$$\sum_{j \in B_k} I_j^{-i} \leq 1 \qquad \forall i,k \in N \tag{13}$$

$$\sum_{b=1}^{a} \delta_{a,b} = 1 \qquad \forall a = 1, ..., m \tag{14}$$

$$\delta_{a,b} \leq \delta_{b,b} \qquad \forall b = 1, ..., m, a = b, ..., m \tag{15}$$

$$\sum_{(a,b) \in \alpha_\phi^*} \delta_{a,b} - I_j \leq |\alpha_\phi^*| \qquad \forall \phi \in \Phi, j \in B_{win(M')} \tag{16}$$

$$I_j, I_j^{-i}, \pi_j(a,b), \pi_j^{-i}(a,b), \Pi_j(a,b), \Pi_j^{-i}(a,b), \delta_{a,b} \in \{0,1\}, p_i \geq 0 \tag{17}$$

Figure 2: Mixed integer program for finding the optimal bundling M^*.

5 Custom branch-and-bound approaches

We will now move on to discussing our custom branch-and-bound approaches. Later we show that these scale significantly better than using a general-purpose MIP solver (CPLEX) on the MIP described in the previous section.

5.1 Branching scheme

To find the optimal bundling we introduce a custom branch-and-bound algorithm, FIND-BUNDLING. It is a tree search algorithm that branches on items. At each node in the search tree, the algorithm branches on an item, with each branch adding the item to a different bundle. One of the branches corresponds to adding it to the empty bundle.

The algorithm explores nodes in best-first order. The revenue obtained from the best solution found so far is a global variable f^*; initially $f^* = 0$. The pseudocode is given in Algorithm 1.

ALGORITHM 1: Custom branch-and-bound algorithm FIND-BUNDLING

Input: The set of items M, the set of bids B
Output: The optimal revenue LB, the optimal bundling

```
1  LB← 0
2  insert ({{CHOOSEBRANCHINGITEM(M, B)}}, ∞) in OPEN      // Open is a priority queue of search tree
   fringe
3                                                          // nodes, sorted in descending order of
4                                                          // the second argument
5  while OPEN not empty do
6      (CURRENT, VAL) ← next in OPEN
7      if UB(CURRENT,M, B)>LB then
8          i ← CHOOSEBRANCHINGITEM(CURRENT,M, B)
9          for b in CURRENT do
10             CHILD ← CURRENT with i added to b
11             f* = MAX(LB(CHILD,M, B),f*)
12             insert (CHILD, UB(CHILD,M, B)) in OPEN
13         end
14         CHILD ← CURRENT with {i} appended to the list of bundles
15         insert (CHILD, UB(CHILD,M, B)) in OPEN
16     end
17 end
```

FIND-BUNDLING starts out with the bundling $\{\{\text{CHOOSEBRANCHINGITEM}(M, B)\}\}$, that is, it chooses an item, and places it in a bundle by itself (Step 2). At each node, the next item i to branch on is chosen (Step 8), and in Step 9-12 FIND-BUNDLING creates a branch for each of the existing bundles, with i added to that bundle. For the last branch in Step 14-15, a new bundle is added with i as the lone item in that bundle. The branching factor at a given node is therefore the number of bundles already created plus one. For each node, the upper bound is used for deciding where in the ordered OPEN list the node is inserted.

$$\max \sum_{j \in B} v_j \cdot I_j \tag{18}$$

$$I_j + I_k \leq 1 \;\; \forall j,k \in B \; s.t. \; S_j \sqcap S_k \neq \emptyset \tag{19}$$

$$I_j + I_k \leq 1 \;\; \forall i \in N, j,k \in B_i \tag{20}$$

$$I_j \in \{0,1\} \tag{21}$$

Figure 3: Mixed integer program for finding the welfare-maximizing allocation for a given bundling b.

5.2 Lower bounding

In Step 11 FIND-BUNDLING computes a lower bound at the node. If a high lower bound is found, we can update f^*, and thereby achieve better pruning.

We use the following technique for lower bounding. For any node v in the search tree, we simulate a VCG auction on the bundles decided on the search path from the root to v along with all the yet-undecided (i.e., yet unbundled) items. Our lower bound is then the sum of the VCG payments from the bidders.

Proposition 5.1. *This is a valid lower bound.*

Proof. One option for FIND-BUNDLING is to take the branch where every item is added in its own separate bundle for all yet-undecided items. This path yields exactly the auction that is used in the lower bound definition. □

In the rest of the paper, whenever we refer to the revenue of a node, we mean the value defined in this section.

To compute the lower bound, we make $n + 1$ calls (one overall and one with each bidder removed in turn) to a subroutine that does (optimal) combinatorial auction winner determination. We call that routine DETERMINE-WINNERS, and give the MIP formulation for it in Figure 3. (In the experiments, we will use a leading general-purpose MIP solver, CPLEX, to solve this MIP, which—while being NP-complete [45, 32]—is dramatically easier in practice than the bundling problem.)

In order to reduce the number of calls to DETERMINE-WINNERS, we reuse the lower bound on revenue from the parent node if the current node is the branch where the item is added alone in a new bundle. This can be done because the MIP at that node is exactly the same as at the parent node.

Typically in tree search/integer programming, if one does not use a lower-bounding technique for the yet-undecided variables, one simply (implicitly) uses a lower bound equal to the value from the variables that have been decided on the path from the root to the current node. For example, in winner determination for combinatorial auctions, one uses the sum of the values of the bids that have been accepted on that path (e.g., [48, 21, 51, 53]). Interestingly, in the bundling setting one needs to be more careful. For example, using just the bids that are only interested in items that

have already been bundled on the path would not give a valid lower bound. The reason is that this could discard a bidder that causes revenue nonmonotonicity (as shown in Section 3.3.)

5.3 Upper bounding

In Steps 7 and 12, FIND-BUNDLING calls a function to upper bound the revenue obtainable in the subtree rooted at the node. In this paper we propose, and investigate the performance of, several such techniques. These techniques are discussed in each of the following subsubsections, respectively. If the technique indeed gives an upper bound (as opposed to sometimes giving a value that is below the actual revenue obtainable in the subtree rooted at the node), that is, the upper-bounding heuristic is *admissible*, FIND-BUNDLING will always give the optimal solution. In addition to admissibility of the different techniques, we will also study their *monotonicity*, that is, whether the upper bound is nonincreasing down each search path.[3]

5.3.1 MAX-WELFARE

The first, and simplest, upper-bounding technique is to use the highest achievable welfare, constrained to honoring the bundling from the path so far. Specifically, the technique generates a set of "items" M' consisting of the *bundles* created so far in the search, and the remaining items unbundled, and then calls DETERMINE-WINNERS on the "item" set. We call this heuristic MAX-WELFARE.

Proposition 5.2. MAX-WELFARE *is admissible and monotonic.*

Proof. We prove monotonicity first. In the computation of MAX-WELFARE, all yet-undecided items at the search node are unbundled. So, MAX-WELFARE corresponds to the welfare of the finest bundling achievable in that subtree. Naturally, the optimal welfare is nondecreasing as we make strictly finer bundlings. It follows that MAX-WELFARE is monotonic.

We prove admissibility next. The welfare at a node upper bounds the revenue at the node. For any descendant d of the current node, we have that MAX-WELFARE at d upper bounds the revenue at d. From monotonicity we have that MAX-WELFARE at the current node is no less than MAX-WELFARE at d. Thus MAX-WELFARE at the current node is an upper bound. □

Note that we could compute the VCG revenue of M', but that would not give an upper bound because VCG revenue can increase with more bundling. That is, a bundling with higher VCG revenue may be found deeper in the current subtree.

5.3.2 EXTERNALITY-FLOW

Our second upper-bounding technique is based on the following proposition.

[3]Unlike in typical tree (or graph) search in artificial intelligence [47], here monotonicity does not imply admissibility because there is no notion of the cost of the path from the root to the node.

Proposition 5.3. *For any bundling in the VCG, the payment for each winning bid j for bidder i can never be more than the welfare of the welfare-maximizing allocation $W^{*,-i}_{\phi,B_{\sqcap j}}$, over the bids $B_{\sqcap j} = \{k \in B_{-i} : S_j \sqcap S_k \neq \emptyset\}$.*

Proof. Proof by contradiction. Suppose this were not the case, and let $p_i = W^{*,-i}_{\phi,B} - W^{*,-i}_{\phi_{-i},B}$ be the payment from i for such an allocation, and let W^{-i}_{ϕ,S_j} be the welfare of the bids that receive items from S_j in $W^{*,-i}_{\phi,B}$. We then have $p_i = W^{*,-i}_{\phi,B} - W^{*,-i}_{\phi_{-i},B} > W^{*,-i}_{\phi,B_{\sqcap j}} \geq W^{-i}_{\phi,S_j} \Leftrightarrow W^{*,-i}_{\phi,B} - W^{-i}_{\phi,S_j} > W^{*,-i}_{\phi_{-i},B}$, which means that the welfare obtained from the items ϕ_{-i} in $W^{*,-i}_{\phi,B}$ is higher than in $W^{*}_{\phi,B}$, contradicting the fact that $W^{*}_{\phi,B}$ is a welfare-maximizing allocation. □

Using Proposition 5.3, we introduce a MIP-based technique that we call EXTERNALITY-FLOW. The basic idea of EXTERNALITY-FLOW is that for every winning bid i in the welfare-maximizing allocation at the current node p, some number of other bids from other bidders do not win because of this bid. We say that they potentially have *externality-flow* to i. Any bid j from a different bidder such that $S_j \sqcap S_i \neq \emptyset$ can automatically send externality-flow to i. Additionally, for any bid k that is not bundling-intersecting with i, if either $S_i \cap M_p \neq \emptyset$ or $S_k \cap M_p \neq \emptyset$, where M_p is the set of items not branched on yet, k can still send externality flow if the upper bound MIP decides to bundle one such unbundled variable in a way that makes the two bids, i and k, bundling-intersecting.

The complete MIP is shown in Figure 4. For all bids j and all winning bids i, we have an externality-flow variable $f_{j,i}$ and a boolean variable $I_{j,i}$ that expresses whether flow is allowed from i to j. For each winning bid i, we have a real-valued payment variable p_i and variables $\gamma^{in}_i, \gamma^{out}_i$ that describe whether the bid can send or receive externality-flow from other winning bids, where only one of the two can be true. Finally, for all $c \in M_p, b \in \phi_{br}$, where ϕ_{br} is bundles already created on the search path, we have a boolean variable $\delta_{c,b}$ that describes whether c is bundled with b, and for all $c, c' \in M_p, b \in \phi_{br}$ we have a boolean variable that describes whether the two unbundled items are sent to the same bundle.

Constraint 23 ensures that the payment is never more than the externality-flow to the bid. Constraint 24 ensures that flow is only sent if the boolean variable allowing flow between the two bids is active. Constraints 25-27 ensure that winning bids only either send or receive flow from other winning bids. Constraint 28, which uses ϕ^i_{br}, the subset of ϕ_{br} intersecting with S_i, ensures that for any two bids that are not bundling-intersecting in the current bundling, flow is only allowed between them if they are made bundling-intersecting through bundling of the remaining unbundled items. Constraint 29 ensures that two unbundled items are only considered intersecting if they are sent to the same bundle. Constraint 30 ensures that each unbundled item is only added to one bundle. Finally, Constraint 32 implements Proposition 5.3, that is, the set of bids that send flow to bidder i must be a valid allocation.

We do not have a proof that this technique upper bounds the revenue found at any node in the subtree, but for all experiments described in Section 6 it worked correctly. Furthermore, we decided not to invest further time in generating a proof because the last two upper-bounding techniques, discussed below, led to better performance anyway in terms of run time of the overall bundling algorithm—due to tighter upper bounding and thus smaller search trees.

$$\max \sum_{i=1}^{n} p_i \tag{22}$$

$$\text{s.t. } p_i \leq \sum_{j \neq i} f_{j,i} \qquad \forall i \in N \tag{23}$$

$$f_{i,j} \leq v_i \cdot I_{i,j} \qquad \forall i,j \in B \tag{24}$$

$$f_{j,i} \leq v_j \cdot \gamma_i^{in} \qquad \forall i,j \in B \; s.t. \; |\alpha_i| > 0, |\alpha_j| > 0 \tag{25}$$

$$f_{i,j} \leq v_i \cdot \gamma_i^{out} \qquad \forall i,j \in B \; s.t. \; |\alpha_i| > 0, |\alpha_j| > 0 \tag{26}$$

$$\gamma_i^{in} + \gamma_i^{out} = 1 \qquad \forall i \in N \; s.t. \; |\alpha_i| > 0 \tag{27}$$

$$I_{i,j} \leq \sum_{b \in \phi_{br}^i} \sum_{c \in M_p^j} \delta_{c,b} + \sum_{c \in M_p^i} \sum_{b \in \phi_{br}^j} \delta_{c,b} + \sum_{c \in M_p^i} \sum_{c' \in M_p^j} \sum_{b \in \phi_{br}} \delta_{c,b,c'} \qquad \forall i,j \in B \; s.t. \; S_j \sqcap S_i = \emptyset \tag{28}$$

$$2\delta_{c,a,c'} - \delta_{c,a} - \delta_{c',a} \leq 0 \qquad \forall c, c' \in M_p, a \in \phi_p \tag{29}$$

$$\sum_{a \in \phi_p} \delta_{c,a} \leq 1 \qquad \forall c \in M_p \tag{30}$$

$$v_i \geq p_i \geq 0, f_{i,j} \geq 0, I_{i,j}, \gamma_i^{in}, \gamma_i^{out}, \delta_{c,a}, \delta_{c,a,c'} \in \{0,1\} \tag{31}$$

$$\sum_{\{j : j \in B, k \in S_j\}} I_{j,i} \leq 1 \qquad \forall i \in B \; s.t. \; |\alpha_i| > 0, k \in M \tag{32}$$

Figure 4: Mixed integer program for EXTERNALITY-FLOW.

5.3.3 VCG$^+$

Our third upper-bounding technique is like computing VCG payments for the bundling at the node but with the negative term chosen so as to maximize payments (under the condition that no bidder pays more than her valuation). We call this technique VCG$^+$.

Let us now formalize this idea. The sum of payments like the VCG payments for the bundling at the node, but with the negative term chosen so as to maximize the payments, is $\max_{\alpha \in A_\phi} \sum_{i \in N} \sum_{j \neq i} [v_j(\alpha^*_{-i}) - v_j(\alpha)]$. Here, $\alpha^*_{-i} \in A_\phi$ is the optimal allocation without bidder i, for the bundling, ϕ, at the node. A_ϕ is the set of allocations consistent with bundling ϕ. Also, $\alpha \in A_\phi$ is any allocation that satisfies the bundling at the node. We further tighten this upper bound by making sure that no bidder is charged more than her valuation. This gives us the formula for our upper-bounding technique VCG$^+$:

$$\max_{\alpha \in A_\phi} \sum_{i \in N} \min \left\{ \sum_{j \neq i} [v_j(\alpha^*_{-i}) - v_j(\alpha)], \; v_i(\alpha) \right\} \tag{33}$$

In the special case where α is the welfare-maximizing allocation at the current node, this equals the revenue of running VCG at the node.

The MIP presented in Figure 5 implements this idea. Constraints 35 and 36 ensure that each

$$\max \sum_{i=1}^{n} p_i \tag{34}$$

$$p_i \leq \sum_{j \in B_i} v_j \cdot I_j \qquad \forall i \in N \tag{35}$$

$$p_i \leq \sum_{j \in B_{-i}} v_j \cdot I_j^{-i} - \sum_{j \in B_{-i}} v_j \cdot I_j \quad \forall i \in N \tag{36}$$

$$\sum_{j \in B_{\sqcap b}} I_j \leq 1 \qquad \forall b \in \phi_p \tag{37}$$

$$\sum_{j \in B_{\sqcap b}} I_j^{-i} \leq 1 \qquad \forall i \in N, b \in \phi_p \tag{38}$$

$$\sum_{j \in B_i} I_j \leq 1 \qquad \forall i \in N \tag{39}$$

$$\sum_{j \in B_k} I_j^{-i} \leq 1 \qquad \forall i, k \in N \tag{40}$$

Figure 5: Mixed integer program for VCG$^+$. Here, $B_{\sqcap b} = \{j \in B | S_j \sqcap b \neq \emptyset\}$ is the set of all bids that are interested in some bundle b.

bidder pays her VCG$^+$payment. Constraints 37 and 38 ensure that only one bidder can win each bundle in ϕ_p, the bundling at the node. Constraints 39 and 40 ensure that each bidder wins only one of his bids.

We now prove that VCG$^+$ gives an upper bound to the revenue found at any node in the subtree rooted at the current node.

Proposition 5.4. VCG$^+$ *is admissible and monotonic.*

Proof. Consider an arbitrary current node p.

We prove admissibility first. VCG$^+$ selects the best set of winning bids from all legal allocations of winning bids for the bundling ϕ_p. For any descendant d, we get a bundling ϕ_d such that if two items $a, b \in M$ are bundled together in ϕ_p then they are also bundled together in ϕ_d. This means that any valid set of winning bids at d is also a valid set of winning bids at p. In particular, the welfare-maximizing allocation at d is a valid set of winning bids at p; call this allocation α_d. Since α_d is a valid allocation for VCG$^+$at p, we just need to show that the payment that VCG$^+$can obtain at p by selecting this allocation is no smaller than $r_d = \sum_{i=1}^{n} [W^{*,-i}_{\phi_d,B} - W^{*,-i}_{\phi_{d,-i},B}]$, the VCG revenue of ϕ_d. This is true because the negative term is the same (since the allocations are the same), and for the positive term $W^{*,-i}_{\phi_d,B}$, we have $W^{*,-i}_{\phi_d} \leq W^{*,-i}_{\phi_p}$ since optimal welfare is nonincreasing with more bundling.

Monotonicity follows from the fact that the VCG$^+$ MIP for any descendant of d of p is the VCG$^+$ MIP for p with additional constraints added. Adding constraints cannot increase the value of the MIP. □

5.3.4 VCG^+_{LB}

In our fourth, final upper-bounding technique, VCG^+_{LB}, we tighten the bound of VCG^+ based on the observation that any constraint that does not cut off any of the welfare-maximizing allocations at any node in the subtree will preserve the upper bound. With this in mind, we add the following constraint to the MIP of Figure 5, where W_{LB} is a lower bound on the welfare found at any node in the subtree.

$$\sum_{j \in B} v_j \cdot I_j \geq W_{LB} \tag{41}$$

At each search node, we use two different ways of computing a value for W_{LB}, and we use the larger of the two. The first is the maximum valuation of any single bid, which is obviously a lower bound since we can always let any single bid be the only winner. The second is obtained by taking all the bundles created so far in the search path, and only auctioning off these bundles, with all items not branched on being unavailable. In other words, we run DETERMINE-WINNERS on the bids that do not use any of the yet-undecided items. This is clearly a lower bound on the welfare of any node in that subtree because the allocation that it finds remains available for selection at all those nodes.

Proposition 5.5. VCG^+_{LB} *is admissible and monotonic.*

Proof. Admissibility follows immediately from the admissibility of VCG^+ and the fact that both ways of computing W_{LB} indeed yield lower bounds as argued above.

Monotonicity follows from monotonicity of VCG^+ and the fact that both ways of computing W_{LB} yield W_{LB} values that are nondecreasing down each search path (because the set of bids to choose from grows or stays the same). □

5.4 Variable ordering

In Step 7, the function CHOOSEBRANCHINGITEM(node) chooses the item to branch on at the node. Any choice will yield a correct algorithm, but some choices lead to smaller tress than others and thus shorter run times. The motivation for our variable-selection heuristic is that we want to pick an item that most likely needs to be bundled so that we get a fairly balanced search tree (where the promising branches are the many branches where this item is bundled). In contrast, branching on an item that likely should not be bundled would render the one "unbundling" branch the most promising and would thus yield lopsided deep trees.

We introduce two branching heuristics, MAX-PRICE-GAP-SIZE and MAX-PRICE-GAP-LOG. They both work by first computing the highest and second-highest "normalized bid price" for each item. Then, the item that has the greatest difference between the highest and second-highest "normalized bid price" is chosen for branching. The idea is that such items are promising for bundling because there is not enough competition on them. MAX-PRICE-GAP-SIZE In MAX-PRICE-GAP-SIZE, we use the following formula for "normalized bid price": $\frac{v_j}{|S_j|}$. In MAX-PRICE-GAP-LOG, we use the following formula for "normalized bid price": $\frac{v_j}{log(|S_j|)}$.

Using the number of items in the bid, $|S_j|$, for normalization gives a more precise estimate of the valuation of each item, but using the logarithm of $|S_j|$ favors bids with a greater number of

items, which can lead to more important decisions being made early. Logarithmic normalization has been experimentally shown to perform well in the winner determination problem [53] so we included that in the experiments in our setting as well.

5.5 Lazy bounding

Finally, we evaluate what we call lazy bounding. The idea is that in Step 12 of FIND-BUNDLING, where UB(node, M, B) is called, we can use the upper bound of the parent for placing the node in the open list—as opposed to computing the actual upper bound at the node.

The disadvantage is that this causes a slightly less precise choice of nodes to pop off of the open list since the list is slightly out of order. Optimality is still maintained by continuing to pop nodes off the open list until the (parent) value of all nodes on the open list is no greater than the value of the best solution found.

The advantage is that we never need to invest the effort to compute the upper bound for nodes that are never popped off the open list. In algorithms like ours, where the branching factor is large ($\Omega(m)$ in our setting), this can have a great impact on performance (especially as the upper bound we compute at each node is a mixed integer program and thereby NP-hard to compute in itself).

6 Experiments

We conducted experiments with all our different algorithmic approaches using the leading combinatorial auction benchmark suite, the Combinatorial Auction Test Suite (CATS) [34]. We generated a test suite from all the CATS distributions (excluding the old, unrealistic "legacy" distributions). So, we had five distributions: arbitrary, matching, paths, regions, and scheduling. For each distribution we generated instances with 4-15 items, and bids equal to 0.2, 0.5, 1, 2, 5, and 10 times the number of items, with 20 instances generated for each of these settings. For space reasons we include only the most interesting results here. For some distribution and bids-to-items multiplier combinations, CATS was unable to generate the desired number of bids for smaller numbers of items, and hence some of our experiments start at a larger number of items than 4 (up to 10 in some cases). All experiments were conducted on a cluster of 7 computers, each with two quad-core AMD Opteron 2.0GHz processors and 32GB of RAM, for a total of 56 cores. Each experiment was run on one thread on one core. The operating system was Rocks Version 6.1. The MIP models were solved using CPLEX 12.5 [29]. We used a time limit of 15 minutes for each run.

In Figures 8, 9, 10, 11, and 12, we present the results of our algorithms on an increasing number of items, where the number of bids generated is five times the number of items.

In Figures 13, 14, 15, 16, and 17, we present the results of our algorithms when keeping the number of items constant at 8 (for all distributions except scheduling, where CATS had to start at 10), while increasing the number of bids.

For each of the figures, the tables are split in three. The top table compares the basic MIP approach to FIND-BUNDLING with all of the special techniques turned off (we denote this by NONE), and FIND-BUNDLING with all of the best special techniques turned on (VCG$^{+}_{LB}$, MAX-PRICE-GAP-LOG, and LAZY-BOUND) (we denote this by MBL-MPGL-L).

In the middle and bottom tables we present experiments on varying, or turning off, one heuristic at a time while keeping the others at their best settings. The middle table presents experiments on FIND-BUNDLING with the best upper bounding technique, VCG^+_{LB}, and varies the other heuristics. The best variant where we use the MAX-PRICE-GAP-LOG and LAZY-BOUND heuristics is not included in this table because it is already presented in the top table. The variant we call MBL-MPGS-L uses the MAX-PRICE-GAP-SIZE and LAZY-BOUND heuristics. The variant we call MBL-MPGL uses the MAX-PRICE-GAP-LOG heuristic. The variant we call MBL-L uses the LAZY-BOUND heuristic.

The bottom table compares the upper-bounding heuristics in FIND-BUNDLING. All of these techniques use MAX-PRICE-GAP-LOG for variable oredering and LAZY-BOUND. Again, the best variant where we use VCG^+_{LB} is not included here because it is already included in the top table. The variant we call MB-MPG-L uses VCG^+. The variant we call EF-MPGL-L uses EXTERNALITY-FLOW. The variant we call MSW-MPGL-L uses MAX-WELFARE.

In the following subsections we discuss the experimental results.

6.1 Basic MIP

For the basic MIP approach, to the left in the top tables, our experiments show that it is unable to scale beyond 7 items for all of the distributions, when generating five times as many bids as items (Figures 8, 9, 10, 11, and 12). This is not surprising, as just the size of the program very rapidly becomes extremely large. When the number of items is kept at 8, while increasing the number of bids, it is able to solve some instances for the arbitrary, regions, paths, and matching distributions (Figures 13, 14, 15, and 16), while it is unable to solve any of the scheduling instances with 10 items, even with only 10 bids (Figure 17), and for all distributions, it does not solve any of the instances with more than 16 bids.

6.2 Upper bounding heuristics

We conducted extensive experiments with each of the four upper-bounding techniques (VCG^+_{LB}, VCG^+, EXTERNALITY-FLOW, and MAX-WELFARE) and with no upper bounding (NONE). For all the tables presented, we kept the other heuristics constant at their best settings: MAX-PRICE-GAP-LOG for variable ordering and LAZY-BOUND (except that for NONE, these techniques were turned off to save run time because they make no difference since every node is expanded when there is no upper bounding). For all figures, we have VCG^+_{LB} in the right of the top tables, NONE in the middle of the top tables, and VCG^+, EXTERNALITY-FLOW, and MAX-WELFARE to the left, middle, and right in the bottom tables, respectively.

For the experiments where we keep the bids-to-items multiplier fixed, on the arbitrary and regions distributions (Figures 8 and 9), we see that VCG^+_{LB} is able to solve at least 4 out of 20 instances for all item sizes, and more than 10 for item sizes 11 and lower. On these distributions, VCG^+ performs almost as well, with only 7 fewer instances solved overall on the arbitrary distribution and 34 fewer on the regions distribution. NONE, EXTERNALITY-FLOW, and MAX-WELFARE all perform significantly worse on these two distributions, with EXTERNALITY-FLOW

solving more than 40 instances less on both distributions, and NONE and MAX-WELFARE both solving about 70 instances less for each distribution.

For the matching distribution (Figure 10), all algorithms have somewhat similar performance, but the relative order of the upper-bounding heuristics is the same. VCG^+_{LB} solving 27 out of 40 instances, whereas NONE solves 20.

For the paths and scheduling distributions (Figures 11 and 12), all the upper-bounding techniques perform significantly worse. VCG^+_{LB} only solves one instance at 12 items and one at 13 items, whereas for other distributions it solved 4 or more instances even at the highest number of items. The relative performance of the upper-bounding techniques remains the same, except that MAX-WELFARE slightly outperforms EXTERNALITY-FLOW on the paths distribution.

For the experiments where we vary the number of bids, all the upper-bounding techniques except NONE solve every instance for the arbitrary, regions, matching, and paths distributions (Figures 8, 9, 10 and 11). VCG^+_{LB} solves up to a factor of 10 faster. For the scheduling distribution (Figure 12), VCG^+_{LB} performs significantly better than the other techniques. It solves 53 out of 60 instances, whereas the other techniques solve 14 or fewer instances, except EXTERNALITY-FLOW. Interestingly, for this distribution, when varying the number of bids, EXTERNALITY-FLOW performs significantly better than all other upper-bounding techniques except VCG^+_{LB}, which is not the case for any other set of experiments.

6.3 Variable ordering heuristics

We experimented with all three settings for the variable ordering heuristic: MAX-PRICE-GAP-LOG, MAX-PRICE-GAP-SIZE, and no variable ordering heuristic. In the figures that we present, the other heuristics are kept constant, using the best settings for the other techniques: VCG^+_{LB} and LAZY-BOUND. For all the figures, we have the MAX-PRICE-GAP-LOG heuristic on the right in the top tables, the MAX-PRICE-GAP-SIZE heuristic on the left in the middle tables, and no heuristic on the right in the middle tables.

When we vary the number of items, the performance on the arbitrary and matching distributions (Figures 8 and 10) is almost the same for all three heuristics, with MAX-PRICE-GAP-LOG solving three more instances than MAX-PRICE-GAP-SIZE. For the regions distribution (Figure 9), we see a more significant performance difference. There, MAX-PRICE-GAP-LOG solves 175 instances, where MAX-PRICE-GAP-SIZE solves 154 instances, and using no heuristic solves 140. Finally, for the paths and scheduling distributions (Figures 11 and 11), MAX-PRICE-GAP-SIZE performs somewhat better than MAX-PRICE-GAP-LOG. On paths, it solves 123 instances while MAX-PRICE-GAP-LOG solves 113. On scheduling, it solves 40 while MAX-PRICE-GAP-LOG solves 35.

For the experiments where we vary the number of bids, there is practically no performance difference for any of the arbitrary, regions, matching, and paths distributions. For the scheduling distribution, however, we see a significant difference, where MAX-PRICE-GAP-LOG and MAX-PRICE-GAP-SIZE perform very similarly, with MAX-PRICE-GAP-SIZE performing slightly better. Using no variable ordering heuristic causes performance to drop significantly: only 14 instances get solved, as opposed to 53 using the heuristics. Interestingly, all 14 solved instances have 50 bids, for which the other two heuristics solve 15 instances each. This suggests that the variable

ordering heuristic is mainly important when the ratio of bids to items is low. One possible explanation for this is that with more bids, one would expect the difference between the normalized first and second price to be lower (especially given that we are using synthetic data), and hence the heuristic item values are close to each other anyway.

6.4 Lazy bounding

We evaluated the performance of LAZY-BOUND, while again using the best settings for the other heuristics: VCG^{+}_{LB} for upper bounding and MAX-PRICE-GAP-LOG for variable ordering. The results are to the right in the top tables, and the results without LAZY-BOUND are in the middle of the middle tables.

For the experiments where we vary the number of bids on the scheduling distribution, we see a significant performance improvement from LAZY-BOUND. It solves 53 out of 60 instances, and only 14 without LAZY-BOUND. On all other experiments, the difference in performance is negligible, with LAZY-BOUND performing slightly better on average runtime, but always lying within one standard deviation of not using LAZY-BOUND.

6.5 Revenue increase and surplus extraction

Finally, we conducted experiments that empirically examine how well our approach can both bridge the gap to the optimal revenue, and how much it improves over the VCG revenue. In Tables 6 and 7 we see the results from running optimal bundling in the VCG, VCG (with no bundling), and welfare computation on each of the five CATS distributions, when the number of bids is 0.5 or 5 times the number of items, respectively. The ratios given are average ratios over 20 instances for each parameter setting.

When using a bids-to-items multiplier of 0.5, for almost all parameter settings, there is at least one instance where VCG yields zero revenue and our optimal bundling in the VCG increases on this revenue. On other parameter settings, the revenue increase ranges from 4% to 139%. For the arbitrary, regions, and matching distributions, optimal bundling yields revenue that is about 50% of the welfare, whereas for the paths and scheduling distributions it achieves significantly less.

For the bids-to-items multiplier 5, the revenue increase over VCG is more modest, and tends to lie in the 2-20% range. This is to be expected because when there is more competition, bundling does not increase revenue as much. We also see that about 90% of the welfare is obtained in revenue for these settings.

For all distributions and bids-to-items multipliers, the fraction of welfare that our technique obtains as revenue tends to increase with the number of items. This may be an artifact of the CATS distributions. They tend to generate some number of bids with high valuations that span almost all the items. With larger numbers of items, this type of bid is more likely to occur since the number of bids is proportional to the number of items. If there are even just two such large high-value bids, they generate enough competition to extract almost all of the welfare as revenue. This also means that in real-life applications, where such bids may be unlikely to occur, the potential revenue increase from using our bundling techniques is higher than these CATS-based experiments suggest.

Another artifact of the CATS distributions is that they generate undominated bids, that is, bids whose value is higher than what could be obtained in welfare from the items in the bid using other bids that only want those items. This means that CATS generates artificially strong competition for the given number of bids and items. This is another reason why one can expect our bundling techniques to increase revenue more in real applications than in the CATS-based experiments.

7 Conclusion and future research

Revenue maximization in combinatorial auctions (and other multidimensional selling settings) is one of the most important and most elusive problems in mechanism design. The design problem is NP-complete, and the optimal designs include features that are not acceptable in many applications, such as favoring some bidders over others and randomization. In this paper, we instead studied a common revenue-enhancement approach—bundling—in the context of the most commonly studied combinatorial auction mechanism, the VCG. A second challenge in mechanism design for combinatorial auctions is that the prior distribution on each bidder's valuation can be doubly exponential. Such priors do not exist in most applications. Rather, in many applications, there is essentially a point prior, which may not be accurate. We adopted the point prior model, and proved robustness to inaccuracy in the prior. Then, we presented a custom branch-and-bound framework for finding the optimal bundling. In that framework, we introduced several techniques for branching, upper bounding, lower bounding, and lazy bounding. Experiments on CATS distributions validated the approach and showed that our techniques dramatically improve scalability over a leading general-purpose MIP solver.

There are many interesting directions for future research. Affine maximizer auctions, virtual valuations combinatorial auctions, and λ-auctions support unlimited, bidder-specific reserve prices. In contrast, we studied bundling alone as a revenue-enhancement tool. There are many interesting questions about the relative power of different forms of bundling and different forms of reserve pricing, and combinations thereof—possibly both in the auction and catalog sales contexts.

Second, we plan to extend our algorithms to settings with structure. For example, in many advertising markets, the inventory segments are defined by vectors of attributes, and that can, in some settings, provide additional structure. We would also like to study cases where items are divisible and/or bids can be accepted partially. Furthermore, we would like to field our approach in real applications.

Bibliography

[1] William James Adams and Janet L. Yellen. Commodity bundling and the burden of monopoly. *Quarterly Journal of Economics*, 90(3):475–498, Aug 1976.

[2] Mark Armstrong. Multiproduct nonlinear pricing. *Econometrica*, 64(1):51–75, Jan 1996.

[3] Mark Armstrong. Optimal multi-object auctions. *Review of Economic Studies*, 67:455–481, 2000.

[4] Lawrence M. Ausubel and Paul Milgrom. The lovely but lonely Vickrey auction. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 1. MIT Press, 2006.

[5] Christopher Avery and Terrence Hendershott. Bundling and optimal auctions of multiple products. *Review of Economic Studies*, 67:483–497, 2000.

[6] Yannis Bakos and Erik Brynjolfsson. Bundling information goods: Pricing, profits, and efficiency. *Management Science*, 45(12):1613–1630, Dec 1999.

[7] Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 29–35, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-236-4. doi: http://doi.acm.org/10.1145/1134707.1134711.

[8] Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Item pricing for revenue maximization. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 50–59, Chicago, IL, 2008.

[9] Michael Benisch and Tuomas Sandholm. A framework for automated bundling and pricing using purchase data. In *Proceedings of the Conference on Auctions, Market Mechanisms and Their Applications (AMMA)*, 2011.

[10] D Berend and T Tassa. Improved bounds on Bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

[11] Chirstopher H. Brooks and Edmund H. Durfee. Toward automated pricing and bundling of information goods. In *AAAI Workshop on Knowledge-based Electronic Markets*, pages 8–13, 2000.

[12] Zümbül Bulut, Ülkü Gürler, and Alper Sen. Bundle pricing of inventories with stochastic demand. *European Journal of Operational Research*, 197(3):897–911, 2009.

[13] Yang Cai, Constantinos Daskalakis, and Matt Weinberg. Optimal multi-dimensional mechanism design: Reducing revenue to welfare maximization. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2012.

[14] I Chakraborty. Bundling decisions for selling multiple objects. *Economic Theory*, 13:723–733, 1999.

[15] Ed H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[16] Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 103–110, Edmonton, Canada, 2002.

[17] Vincent Conitzer and Tuomas Sandholm. Applications of automated mechanism design. In *UAI-03 workshop on Bayesian Modeling Applications*, Acapulco, Mexico, 2003.

[18] Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 132–141, New York, NY, 2004.

[19] Robert E. Dansby and Cecilia Conrad. Commodity bundling. *The American Economic Review*, 74(2): 377–381, May 1984.

[20] Yuval Emek, Michal Feldman, Iftah Gamzu, Renato Paes Leme, and Moshe Tennenholtz. Signaling schemes for revenue maximization. In *ACM Conference on Electronic Commerce*, pages 514–531, 2012.

[21] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

[22] G. Gallego and G. van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, 45(1):24–41, 1997.

[23] Arpita Ghosh, Hamid Nazerzadeh, and Mukund Sundararajan. Computing optimal bundles for sponsored search. In *International Workshop On Internet And Network Economics (WINE)*, pages 576–583, 2007.

[24] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[25] Joseph P. Guiltinan. The price bundling of services: A normative framework. *Journal of Marketing*, 51(2):74–85, Apr 1987.

[26] Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1164–1173, 2005.

[27] Ward Hanson and R. Kipp Martin. Optimal bundle pricing. *Manage. Sci.*, 36(2):155–174, 1990. ISSN 0025-1909.

[28] Lorin M. Hitt and Pei-yu Chen. Bundling with customer self-selection: A simple approach to bundling low-marginal-cost goods. *Management Science*, 51(10):1481–1493, 2005.

[29] IBM ILOG. CPLEX 12.5 User's Manual, 2012.

[30] Kamel Jedidi, Sharan Jagpal, and Puneet Manchanda. Measuring heterogeneous reservation prices for product bundles. *Marketing Science*, 22(1):107–130, 2003.

[31] Philippe Jehiel, Moritz Meyer-Ter-Vehn, and Benny Moldovanu. Mixed bundling auctions. *Journal of Economic Theory*, 127(1):494–512, 2007.

[32] Richard Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972.

[33] Jeffrey O. Kephart, Christopher H. Brooks, Rajarshi Das, Jeffrey K. MacKie-Mason, Robert Gazzale, and Edmund H. Durfee. Pricing information bundles in a dynamic environment. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 180–190, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-387-1. doi: http://doi.acm.org/10.1145/501158.501178.

[34] Kevin Leyton-Brown, Mark Pearson, and Yoav Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 66–76, Minneapolis, MN, 2000.

[35] Anton Likhodedov and Tuomas Sandholm. Methods for boosting revenue in combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 232–237, San Jose, CA, 2004.

[36] Anton Likhodedov and Tuomas Sandholm. Approximating revenue-maximizing combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Pittsburgh, PA, 2005.

[37] A Manelli and D Vincent. Bundling as an optimal selling mechanism for a multiple-good monopolist. *Journal of Economic Theory*, 127(1):1–35, 2006.

[38] Eric Maskin and John Riley. Optimal multi-unit auctions. In Frank Hahn, editor, *The Economics of Missing Markets, Information, and Games*, chapter 14, pages 312–335. Clarendon Press, Oxford, 1989.

[39] R. Preston McAfee, John McMillan, and Michael D. Whinston. Multiproduct monopoly, commodity bundling, and correlation of values. *Quarterly Journal of Economics*, 104(2):371–383, May 1989.

[40] Peter Bro Miltersen and Or Sheffet. Send mixed signals: earn more, work less. In *ACM Conference on Electronic Commerce*, pages 234–247, 2012.

[41] Roger Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

[42] George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999.

[43] Thomas Palfrey. Bundling decisions by a multiproduct monopolist with incomplete information. *Econometrica*, 51:463–484, 1983.

[44] Baharak Rastegari, Anne Condon, and Kevin Leyton-Brown. Revenue monotonicity in deterministic, dominant-strategy combinatorial auctions. *Artif. Intell.*, 175(2):441–456, 2011.

[45] Michael Rothkopf, Aleksandar Pekeč, and Ronald Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[46] Paat Rusmevichientong, Benjamin Van Roy, and Peter W. Glynn. A nonparametric approach to multiproduct pricing. *Oper. Res.*, 54(1):82–98, 2006. ISSN 0030-364X. doi: http://dx.doi.org/10.1287/opre.1050.0252.

[47] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.

[48] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002.

[49] Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002. Earlier versions: ICE-98 keynote; WUCS-99-01 Jan. 1999; IJCAI-99.

[50] Tuomas Sandholm. Automated mechanism design: A new application area for search algorithms. In *International Conference on Principles and Practice of Constraint Programming (CP)*, pages 19–36, Cork, Ireland, 2003.

[51] Tuomas Sandholm and Subhash Suri. BOB: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence*, 145:33–58, 2003.

[52] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Anytime coalition structure generation with worst case guarantees. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 46–53, Madison, WI, July 1998.

[53] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, 2005.

[54] Richard Schmalensee. Gaussian demand and commodity bundling. *The Journal of Business*, 57(1): S211–S230, Jan 1984.

[55] George J. Stigler. United States v. Loew's Inc.: A note on block-booking. *The Supreme Court Review*, 1963:152–157, 1963.

[56] Kalyan T. Talluri and Garrett John Van Ryzin. *The theory and practice of revenue management*. Number 68 in International series in operations research and management science. Kluwer Academic Publisher, Boston, Mass. [u.a.], 2004.

[57] R. Venkatesh and W. Kamakura. Optimal bundling and pricing under a monopoly: Contrasting complements and substitutes from independently valued products. *Journal of Business*, 76(2):211–231, 2003.

[58] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[59] William Walsh, Craig Boutilier, Tuomas Sandholm, Robert Shields, George Nemhauser, and David Parkes. Automated channel abstraction for advertising auctions. In *Proceedings of the Ad Auctions Workshop*, 2009.

[60] Shin-yi Wu, Lorin M. Hitt, Pei-yu Chen, and G. Anandalingam. Customized bundle pricing for information goods: A nonlinear mixed-integer programming approach. *Management Science*, 54(3): 608–622, 2008.

APPENDIX

#items	arbitrary-0.5	regions-0.5	matching-0.5	paths-0.5	scheduling-0.5
4	∞ / 0.3717	∞ / 0.4986	- / -	- / -	- / -
5	∞ / 0.3832	∞ / 0.2920	- / -	∞ / 0.0115	- / -
6	∞ / 0.6545	∞ / 0.4897	- / -	∞ / 0.1072	- / -
7	1.0645 / 0.7130	∞ / 0.5704	- / -	∞ / 0.1586	- / -
8	∞ / 0.6572	∞ / 0.4275	∞ / 0.4269	∞ / 0.1093	- / -
9	∞ / 0.5133	∞ / 0.4985	- / -	∞ / 0.1597	- / -
10	∞ / 0.4084	∞ / 0.6559	- / -	∞ / 0.3345	∞ / 0.2736
11	∞ / 0.5804	1.0401 / 0.7726	- / -	∞ / 0.3848	∞ / 0.2873
12	∞ / 0.4525	∞ / 0.6189	∞ / 0.5384	∞ / 0.3733	∞ / 0.1459
13	∞ / 0.4548	∞ / 0.5035	- / -	∞ / 0.4526	∞ / 0.1167
14	1.8161 / 0.6212	2.3901 / 0.5410	- / -	∞ / 0.3987	∞ / 0.1906
15	∞ / 0.5751	∞ / 0.5271	- / -	∞ / 0.4215	∞ / 0.2560

Figure 6: Ratios of the revenue of the optimal abstraction, r_{OPT}, to the VCG revenue, r_{VCG}, and the social welfare, W^*_M, for varying numbers of items for the CATS distributions, with a bid multiplier of 0.5. The first number in each column is $\frac{r_{OPT}}{r_{VCG}}$ and the second is $\frac{r_{OPT}}{W^*_M}$. ∞ means that VCG achieved 0 revenue, where optimal bundling received more.

#items	arbitrary-5	regions-5	matching-5	paths-5	scheduling-5
4	1.0587 / 0.8942	1.0456 / 0.9051	- / -	- / -	- / -
5	1.0159 / 0.8990	1.0758 / 0.8731	- / -	1.0055 / 0.8940	- / -
6	1.0441 / 0.8995	1.2002 / 0.8211	- / -	1.0169 / 0.8563	- / -
7	1.0627 / 0.8705	1.2013 / 0.8399	- / -	1.0763 / 0.8521	- / -
8	1.0895 / 0.8786	1.1163 / 0.8357	1.0343 / 0.8428	1.0224 / 0.8665	- / -
9	1.0259 / 0.9232	1.0465 / 0.9135	- / -	1.0387 / 0.8752	- / -
10	1.0219 / 0.9108	1.0413 / 0.9168	- / -	1.0122 / 0.8878	1.0542 / 0.8751
11	1.0376 / 0.9195	1.0979 / 0.9207	- / -	1.0304 / 0.9062	1.0508 / 0.8826
12	1.0855 / 0.9151	1.0356 / 0.9220	1.0841 / 0.8910	1.0202 / 0.9149	1.0297 / 0.8649
13	1.0727 / 0.9485	1.0929 / 0.9252	- / -	1.0022 / 0.9338	1.0589 / 0.9339
14	1.0338 / 0.9233	1.0000 / 0.9703	- / -	- / -	1.0358 / 0.8404
15	1.0000 / 0.9628	- / -	- / -	- / -	1.0731 / 0.8933

Figure 7: Ratios of the revenue of the optimal abstraction, r_{OPT}, to the VCG revenue, r_{VCG}, and the social welfare, W^*_M, for varying numbers of items for the CATS distributions, with a bid multiplier of 5. The first number in each column is $\frac{r_{OPT}}{r_{VCG}}$ and the second is $\frac{r_{OPT}}{W^*_M}$.

# items	MIP			NONE			MBL-MPGL-L		
4	10.48	2.74	20	0.79	0.15	20	0.37	0.25	20
5	91.89	59.59	20	3.02	0.75	20	0.77	0.67	20
6	608.47	180.54	17	12.37	2.41	20	2.09	2.27	20
7	746.31	120.76	2	52.43	10.38	20	5.73	6.69	20
8	-	-	0	256.71	58.56	20	17.66	24.31	20
9	-	-	0	-	-	0	90.76	112.85	20
10	-	-	0	-	-	0	103.42	233.66	20
11	-	-	0	-	-	0	160.50	181.30	12
12	-	-	0	-	-	0	181.09	237.07	10
13	-	-	0	-	-	0	55.15	100.91	8
14	-	-	0	-	-	0	124.37	173.56	3
15	-	-	0	-	-	0	0.51	0.24	5
total	-	-	59	-	-	100	-	-	178
# items	MBL-MPGS-L			MBL-MPGL			MBL-L		
4	0.36	0.24	20	0.43	0.31	20	0.38	0.24	20
5	0.90	0.87	20	0.93	0.87	20	0.81	0.76	20
6	2.50	2.62	20	2.71	3.07	20	2.83	3.16	20
7	5.68	8.43	20	8.04	9.91	20	7.52	8.85	20
8	24.70	30.55	20	25.56	37.45	20	26.03	29.10	20
9	81.81	75.55	20	138.11	174.05	20	101.94	92.50	20
10	57.95	131.31	18	83.08	210.55	19	86.34	217.51	19
11	240.24	305.00	12	201.31	230.38	12	224.17	237.16	12
12	229.31	320.22	9	225.05	301.71	10	301.54	371.07	10
13	145.54	264.14	8	73.69	134.67	8	100.22	179.45	8
14	80.06	79.36	2	185.18	259.21	3	252.70	351.06	3
15	19.30	41.81	6	0.52	0.25	5	0.53	0.19	5
total	-	-	175	-	-	177	-	-	177
# items	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
4	0.41	0.26	20	0.37	0.24	20	0.91	0.20	20
5	0.90	0.79	20	1.21	1.03	20	3.36	0.92	20
6	2.61	2.58	20	3.41	2.96	20	13.27	2.58	20
7	7.87	9.39	20	14.90	12.70	20	46.59	11.84	20
8	30.78	36.52	20	71.21	59.50	20	251.27	50.44	20
9	123.63	137.07	20	284.66	237.96	20	605.44	205.89	7
10	46.52	163.46	17	0.51	0.52	11	-	-	0
11	134.32	172.75	10	0.77	0.81	5	-	-	0
12	237.21	336.51	9	0.46	0.15	4	-	-	0
13	79.76	137.77	8	-	-	0	-	-	0
14	2.04	1.48	2	-	-	0	-	-	0
15	0.59	0.11	5	-	-	0	-	-	0
total	-	-	171	-	-	140	-	-	107

Figure 8: Performance characteristics over varying numbers of items for the arbitrary distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# items	MIP			NONE			MBL-MPGL-L		
4	10.21	4.35	20	0.85	0.20	20	0.65	0.41	20
5	166.07	100.97	20	3.48	0.63	20	2.75	0.81	20
6	811.83	111.40	3	17.12	3.06	20	8.95	3.36	20
7	-	-	0	85.05	15.73	20	13.96	8.42	20
8	-	-	0	429.32	80.50	20	27.18	17.17	20
9	-	-	0	874.36	12.46	2	31.39	51.10	20
10	-	-	0	-	-	0	93.76	175.60	19
11	-	-	0	-	-	0	170.20	254.27	14
12	-	-	0	-	-	0	104.60	99.73	5
13	-	-	0	-	-	0	193.33	235.65	9
14	-	-	0	-	-	0	261.35	338.03	4
15	-	-	0	-	-	0	328.77	275.69	4
total	-	-	43	-	-	102	-	-	175
# items	MBL-MPGS-L			MBL-MPGL			MBL-L		
4	0.67	0.44	20	0.82	0.54	20	0.65	0.40	20
5	2.47	0.73	20	3.74	1.22	20	3.35	0.95	20
6	9.05	2.03	20	11.50	4.58	20	15.84	2.54	20
7	25.17	11.17	20	17.07	11.42	20	42.39	15.76	20
8	75.03	32.60	20	33.48	22.44	20	122.73	45.05	20
9	48.49	60.33	20	43.79	74.23	20	88.39	133.52	20
10	209.93	270.04	20	74.63	109.94	18	193.94	231.87	16
11	508.73	308.55	8	150.38	233.92	13	9.71	9.55	2
12	23.11	5.18	3	124.66	129.67	5	-	-	0
13	34.77	17.14	2	137.46	158.13	8	65.23	0.00	1
14	0.31	0.00	1	74.79	76.17	3	0.27	0.00	1
15	-	-	0	365.92	310.72	4	-	-	0
total	-	-	154	-	-	171	-	-	140
# items	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
4	0.69	0.44	20	0.66	0.47	20	1.02	0.25	20
5	2.94	0.89	20	3.50	0.75	20	3.75	0.64	20
6	10.86	3.14	20	16.70	3.75	20	15.54	3.24	20
7	21.80	10.08	20	55.94	23.75	20	64.45	9.31	20
8	62.19	28.35	20	223.32	57.37	20	220.86	54.18	20
9	72.03	87.75	20	224.34	244.97	20	692.65	176.88	6
10	122.22	160.84	16	152.33	287.86	9	-	-	0
11	160.48	217.05	3	0.34	0.00	1	-	-	0
12	-	-	0	-	-	0	-	-	0
13	2.61	0.00	1	-	-	0	-	-	0
14	0.34	0.00	1	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	141	-	-	130	-	-	106

Figure 9: Performance characteristics over varying numbers of items for the regions distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# items	MIP			NONE			MBL-MPGL-L		
8	-	-	0	495.74	65.32	20	38.24	19.60	20
12	-	-	0	-	-	0	507.84	146.89	7
total	-	-	0	-	-	20	-	-	27
# items	MBL-MPGS-L			MBL-MPGL			MBL-L		
8	38.86	20.06	20	46.84	25.11	20	30.50	16.18	20
12	515.05	151.54	7	513.13	74.09	6	722.19	78.68	3
total	-	-	27	-	-	26	-	-	23
# items	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
8	38.01	20.98	20	143.25	61.57	20	129.54	56.86	20
12	533.55	103.62	6	-	-	0	-	-	0
total	-	-	26	-	-	20	-	-	20

Figure 10: Performance characteristics over varying numbers of items for the matching distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# items	MIP			NONE			MBL-MPGL-L		
5	63.44	55.13	20	5.84	0.95	20	4.02	0.70	20
6	326.52	196.45	20	29.00	5.86	20	12.88	7.01	20
7	753.35	91.69	4	152.03	22.82	20	37.59	29.85	20
8	-	-	0	781.62	117.91	16	128.51	159.80	20
9	-	-	0	-	-	0	283.12	96.71	17
10	-	-	0	-	-	0	518.51	259.06	10
11	-	-	0	-	-	0	476.97	211.32	5
12	-	-	0	-	-	0	793.18	0.00	1
13	-	-	0	-	-	0	-	-	0
14	-	-	0	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	44	-	-	76	-	-	113
# items	MBL-MPGS-L			MBL-MPGL			MBL-L		
5	3.66	0.70	20	4.77	0.98	20	3.80	0.69	20
6	10.52	6.06	20	15.16	8.24	20	11.77	7.01	20
7	26.67	22.95	20	45.10	36.83	20	33.99	28.74	20
8	108.56	186.20	20	155.45	196.77	20	118.08	152.12	20
9	193.88	102.56	17	332.66	120.01	17	267.46	96.06	17
10	392.23	200.23	16	539.75	278.61	9	524.55	270.88	11
11	378.21	178.33	7	427.96	89.78	4	487.33	203.58	5
12	363.02	0.00	1	858.09	0.00	1	792.10	0.00	1
13	745.41	141.81	2	-	-	0	867.22	0.00	1
14	-	-	0	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	123	-	-	111	-	-	115
# items	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
5	3.70	0.69	20	5.05	0.65	20	4.37	0.77	20
6	12.22	7.24	20	13.72	4.15	20	14.69	6.22	20
7	33.48	29.44	20	43.59	22.36	20	44.38	29.91	20
8	114.90	155.77	20	157.21	59.94	20	184.72	148.94	20
9	240.50	80.63	17	517.13	224.05	13	511.72	250.83	15
10	504.43	241.79	13	493.93	268.05	3	634.07	273.31	6
11	487.76	254.28	6	-	-	0	837.36	0.00	1
12	689.76	0.00	1	-	-	0	-	-	0
13	724.36	0.00	1	-	-	0	-	-	0
14	-	-	0	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	118	-	-	96	-	-	102

Figure 11: Performance characteristics over varying numbers of items for the paths distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# items	MIP			NONE			MBL-MPGL-L		
10	-	-	0	-	-	0	116.20	237.27	15
11	-	-	0	-	-	0	268.38	283.65	12
12	-	-	0	-	-	0	198.90	251.02	7
13	-	-	0	-	-	0	8.95	0.00	1
14	-	-	0	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	0	-	-	0	-	-	35
# items	MBL-MPGS-L			MBL-MPGL			MBL-L		
10	110.95	212.07	15	85.08	167.90	14	157.68	246.94	14
11	58.70	118.51	12	342.56	345.25	12	180.30	226.79	8
12	347.59	321.53	11	259.04	333.26	7	151.66	250.43	4
13	265.65	262.40	2	29.09	0.00	1	8.22	0.00	1
14	-	-	0	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	40	-	-	34	-	-	27
# items	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
10	114.02	175.94	14	143.62	226.60	7	-	-	0
11	233.21	320.67	8	0.43	0.00	1	-	-	0
12	144.20	239.01	4	-	-	0	-	-	0
13	9.88	0.00	1	-	-	0	-	-	0
14	-	-	0	-	-	0	-	-	0
15	-	-	0	-	-	0	-	-	0
total	-	-	27	-	-	8	-	-	0

Figure 12: Performance characteristics over varying numbers of items for the scheduling distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# bids	MIP			NONE			MBL-MPGL-L		
8	93.70	15.61	20	151.61	15.31	20	0.06	0.02	20
16	249.73	171.14	19	162.90	32.76	20	5.94	12.21	20
40	-	-	0	256.71	58.56	20	17.66	24.31	20
80	-	-	0	361.56	60.90	20	28.77	41.23	20
total	-	-	39	-	-	80	-	-	80
# bids	MBL-MPGS-L			MBL-MPGL			MBL-L		
8	0.06	0.02	20	0.06	0.02	20	0.08	0.06	20
16	10.32	28.11	20	8.34	17.25	20	10.12	32.43	20
40	24.70	30.55	20	25.56	37.45	20	26.03	29.10	20
80	39.12	41.30	20	45.55	76.28	20	31.18	34.67	20
total	-	-	80	-	-	80	-	-	80
# bids	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
8	0.08	0.03	20	20.95	27.76	20	170.62	15.75	20
16	12.50	30.80	20	25.09	31.97	20	181.70	34.93	20
40	30.78	36.52	20	71.21	59.50	20	251.27	50.44	20
80	40.82	55.53	20	108.69	94.23	20	234.50	73.04	20
total	-	-	80	-	-	80	-	-	80

Figure 13: Performance characteristics over varying numbers of bids for the arbitrary distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# bids	MIP			NONE			MBL-MPGL-L		
8	158.36	56.44	20	185.83	35.37	20	13.34	15.39	20
16	430.72	239.91	7	264.16	79.97	20	28.88	20.65	20
40	-	-	0	429.32	80.50	20	27.18	17.17	20
80	-	-	0	552.74	98.31	20	212.12	103.34	20
total	-	-	27	-	-	80	-	-	80
# bids	MBL-MPGS-L			MBL-MPGL			MBL-L		
8	18.27	15.59	20	14.43	15.20	20	47.62	40.47	20
16	50.44	32.43	20	37.36	29.44	20	76.33	45.28	20
40	75.03	32.60	20	33.48	22.44	20	122.73	45.05	20
80	172.20	85.86	20	283.80	143.26	20	290.92	86.11	20
total	-	-	80	-	-	80	-	-	80
# bids	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
8	28.80	23.13	20	113.02	50.45	20	169.26	36.34	20
16	54.97	28.78	20	179.90	81.79	20	210.28	45.01	20
40	62.19	28.35	20	223.32	57.37	20	220.86	54.18	20
80	220.82	109.06	20	386.84	110.35	20	249.24	67.07	20
total	-	-	80	-	-	80	-	-	80

Figure 14: Performance characteristics over varying numbers of bids for the regions distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# bids	MIP			NONE			MBL-MPGL-L		
8	240.61	106.41	20	262.61	34.76	20	24.84	11.06	20
16	489.58	206.26	11	358.20	70.36	20	54.89	29.32	20
40	-	-	0	495.74	65.32	20	38.24	19.60	20
80	-	-	0	567.70	72.31	20	35.90	17.76	20
total	-	-	31	-	-	80	-	-	80
# bids	MBL-MPGS-L			MBL-MPGL			MBL-L		
8	24.86	10.71	20	28.97	13.01	20	42.03	16.85	20
16	55.82	30.47	20	72.89	42.69	20	66.75	42.04	20
40	38.86	20.06	20	46.84	25.11	20	30.50	16.18	20
80	36.25	18.37	20	43.83	26.88	20	32.50	13.05	20
total	-	-	80	-	-	80	-	-	80
# bids	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
8	70.29	33.07	20	210.79	47.61	20	228.85	38.38	20
16	74.41	39.56	20	268.30	64.36	20	253.32	62.18	20
40	38.01	20.98	20	143.25	61.57	20	129.54	56.86	20
80	32.79	17.20	20	118.50	65.57	20	74.45	33.23	20
total	-	-	80	-	-	80	-	-	80

Figure 15: Performance characteristics over varying numbers of bids for the matching distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# bids	MIP			NONE			MBL-MPGL-L		
8	537.58	262.93	15	329.42	67.20	20	48.96	37.83	20
16	691.89	83.43	2	594.38	96.82	20	196.88	120.90	20
40	-	-	0	781.62	117.91	16	128.51	159.80	20
total	-	-	17	-	-	56	-	-	60
# bids	MBL-MPGS-L			MBL-MPGL			MBL-L		
8	84.85	65.83	20	62.06	49.35	20	64.74	37.68	20
16	160.71	116.34	20	242.63	150.85	20	153.77	85.81	20
40	108.56	186.20	20	155.45	196.77	20	118.08	152.12	20
total	-	-	60	-	-	60	-	-	60
# bids	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
8	241.78	94.71	20	375.83	92.47	20	354.61	70.92	20
16	212.76	128.48	20	454.85	95.12	20	449.85	108.35	20
40	114.90	155.77	20	157.21	59.94	20	184.72	148.94	20
total	-	-	60	-	-	60	-	-	60

Figure 16: Performance characteristics over varying numbers of bids for the paths distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.

# bids	MIP			NONE			MBL-MPGL-L		
10	-	-	0	-	-	0	24.41	65.93	20
20	-	-	0	-	-	0	80.76	189.60	18
50	-	-	0	-	-	0	116.20	237.27	15
total	-	-	0	-	-	0	-	-	53
# bids	MBL-MPGS-L			MBL-MPGL			MBL-L		
10	34.29	83.82	20	-	-	0	-	-	0
20	29.25	55.05	18	-	-	0	-	-	0
50	110.95	212.07	15	85.08	167.90	14	157.68	246.94	14
total	-	-	53	-	-	14	-	-	14
# bids	MB-MPGL-L			EF-MPGL-L			MSW-MPGL-L		
10	-	-	0	177.10	278.46	13	-	-	0
20	-	-	0	179.39	253.61	14	-	-	0
50	114.02	175.94	14	143.62	226.60	7	-	-	0
total	-	-	14	-	-	34	-	-	0

Figure 17: Performance characteristics over varying numbers of bids for the scheduling distribution. The first two columns for each algorithm are average runtimes and standard deviation on the solved instances, while the last column lists how many of the 20 instances were solved. Finally, the last row gives the total number of instances solved for each algorithm.