Hybrid Programmatic TV Markets

Problem Presenter

Marco Montes de Oca, Clypd

Report Editor

David A. Edwards, University of Delaware

Thirty-Second Annual Workshop on Mathematical Problems in Industry
June 13–17, 2016
Duke University

Table of Contents

Preface

At the 32nd Annual Workshop on Mathematical Problems in Industry (MPI), Marco Montes de Oca of Clypd presented a problem concerning the optimization of auctions of television advertising slots.

This manuscript is really a collection of reports from teams in the group working on several aspects of the problem. Here is a brief summary of each:

1. Edwards wrote up the main results that the bulk of the group generated during the week. His chapter contains an outline of the general problem, and presents results from some simple optimization algorithms.
2. Emerick and Goldwyn analyze the "max-sum" and "sum-max" no-replacement algorithms and discuss the effect of increasing pod size on total number of placements.
3. Vazquez and Palmer discuss other ways to weight the value matrix, other ways to provide counteroffers and discounts, and how to give preference to certain buyers.
4. Sirlanci provides a flow chart outlining the entire algorithm.
5. Narayanan outlines an award algorithm based upon satisfying as many complete bids as possible.
6. Chugunova illustrates a paradox regarding the value matrix whereby a bidder can increase his chances of winning a bid by reducing the number of slots in the bid.
7. Teresa and Vazquez discuss various ways to provide replacement bids to losing bidders.

In addition to the authors of these reports, the following people participated in the group discussions:

- Joseph Fehribach, Worcester Polytechnic Institute
- Azmat Hussain, North Carolina State University
- Tuan Le, George Mason University
- Qingxia Lia, Fisk University
- Christopher Raymond, University of Delaware

Special recognition is due to Marylin Vazquez, Erik Palmer, and Melike Sirlanci for making the group's oral presentations throughout the week.

Model Formulation; Simple Algorithms

David A. Edwards, University of Delaware

NOTE: Though Edwards wrote up this chapter, it is a summary of the week's work of the entire group (enumerated in the preface).

Section 1: Introduction

Advertising agencies want to place television advertisements for their clients which will expose their products to a particular target audience. The target may consist of characteristics such as age, gender, income level, etc. Typically an advertiser would like a certain amount of *impressions* (*i.e.*, views) for each advertising campaign, either for the target audience or the total audience. These impressions are further categorized as *total impressions* (number of times the ad was seen) and *unique impressions* (*i.e.*, the number of *different* people who saw the ad).

Agencies (*buyers*) place orders with the networks (*sellers*) for desired commercial *slots*. A slot is a designated time on a particular network (*e.g.*, CNN on September 1, 2016, from 10:12–10:14 am) which can run one or more commercials in *pods*. (For instance, the slot described above could contain four 30-second pods.) The orders come in two types:

- *constrained orders*, where the order specifies a certain number of impressions for a particular target audience. It is then the responsibility of the seller to assign the commercials to specific slots which historical data indicate will provide the target audience. (Consideration of this facet of the problem is presented in [1].)
- *bid orders*, where the buyers (having presumably done their own market research) list specific slots (time, date, and network).

Typically bid orders will pay a premium for their slots due to their selectivity. The remaining slots are then assigned to the constrained orders while matching the target audience characteristics.

Often, more than one bid order will come in for the same slot. In that case, the sellers cannot completely fill every order. Rather, they must make a *counteroffer* for an incomplete order. The counteroffer can consist of one or more of the following:

- an offer to run a subset of the slots requested, at a reduced rate. Not only is the rate reduced by the number of slots not filled, but an additional discount is often taken since buyers are typically willing to pay a premium for a specific series of slots.
- replacement of the unfulfilled slots with other slots, which the seller can hopefully justify as having the same sort of target audience as the unfulfilled slots.

The default option is at the least to provide replacements, with or without discount. In addition, typically the largest advertisers are given priority on receiving complete bids.

The above discussion refers to a *synchronous auction*, where all bids come in at the same time, and are settled at the same time. Often advertising auctions are *asynchronous*, and bids come in at various times. As a simple example, consider a week-long auction where bids are due each day at noon. Each evening, the sellers provide feedback to the buyers (information about counteroffers, etc.) given all bids submitted to date. This process is repeated for a week, and then final counteroffers are submitted and trades are settled.

Fortunately, given that settlement issues do not figure heavily in the subsequent anal-

ysis, we choose to treat a week-long asynchronous auction as a series of seven synchronous ones.

In this manuscript, we study several such auction scenarios so that the sellers may maximize their revenue.

Section 2: The Bid-Order Auction

We begin by considering the case of bid orders only, especially since the work in [1] gives a way to convert constrained orders into bid orders.

Let $j_{\max}$ be the number of slots available. (In a typical real-world scenario, $j_{\max}$ can be $O(10^5)$; for the test data we used at the workshop, we had $j_{\max}$ =14,861, which corresponds to around a week's worth of slots.) For each slot, we had viewership data (since the test data was historical). In particular, we knew how many impressions (viewers) each slot had, by gender and by age bracket. This data can be expressed as a 3-tensor V, the *viewership tensor*, where

$$v_{jag} = \text{impressions for age bracket } a \text{ and gender indicator variable } g \text{ for spot } j. \quad (2.1)$$

The interpretations of a and g are given in Table 2.1. Note that there are 30 different demographic categories for each slot.

a	Age Range	a	Age Range	g	Gender
1	2–5	9	30–34	1	Female
2	6–8	10	35–39	2	Male
3	9–11	11	40–44		
4	12–14	12	45–49		
5	15–17	13	50–54		
6	18–20	14	55–64		
7	21–24	15	65+		
8	25–29				

Table 2.1. Interpretation of a and g.

Let $i_{\max}$ be the number of bid orders submitted, which is typically on the order of hundreds. Typically each advertiser submits only one bid. The bid consists of two parts:

1. a list of slots the advertiser wants. The information from all the orders can be compiled into a *bid matrix* B, where

$$b_{ij} = \begin{cases} 1, & \text{if order } i \text{ requests slot } j, \\ 0, & \text{else.} \end{cases} \quad (2.2)$$

   Bids typically request only a few percent of the slots available, so B will be sparse. Nevertheless, there are $2^{j_{\max}}$ different possible combinations for the orders (with replacements), so treating the problem as some sort of integer programming problem can quickly become unwieldy.
2. a *bid price* u_i the advertiser is willing to pay for the entire order submitted.

We did not have access to actual bid data. Instead, we simulated the bids as follows:

1. We created a series of impression requirements modeling reasonable desires of advertisers.

2. We then used the historical test data (where V is known) to generate bids that would satisfy those constraints.

These then became the bids for our optimization procedure.

Each slot j has a *reserve price* r_j, which reflects the lowest price for which a seller is willing to sell a slot. This then establishes a lower bound for the bids:

$$\sum_j b_{ij} r_j \leq u_i. \tag{2.3}$$

Hence any bid for which (2.3) is not satisfied should be rejected. However, we will see in the next section that there are ways to salvage such bids and give the buyer some subset of what (s)he wants. Also, in some cases (say, when a slot will air before the next auction), the seller may be willing to sell the spot for less than r_j rather than for it to remain unsold.

The seller's objective is to maximize the revenue received. If there were no overlap between the bids, the seller would automatically accept any bid for which (2.3) is satisfied, since no other seller would want the same slots. Similarly, if there were perfect overlap between the bids, for each group of slots the seller would accept the largest bid that satisfied (2.3).

Unfortunately, there is often much (but not perfect) overlap between the bids. Hence a counteroffer must be proffered. The seller's counteroffer consists of

1. an *assignment matrix* C, where

$$c_{ij} = \begin{cases} 1, & \text{if order } i \text{ is given slot } j, \\ 0, & \text{else.} \end{cases} \tag{2.4}$$

   Note that the number of ones in column j is at most the number of pods in slot j, which we shall denote by π_j. Hence

$$\sum_i c_{ij} \leq \pi_j, \quad j = 1, \ldots, j_{\max}, \tag{2.5}$$

   and C will be even more sparse than B.
2. a *counteroffer price* x_i the seller is willing to accept for the slot assignment.

   At this stage, the seller then wants to maximize the sellers' revenue f:

$$f = \sum_i x_i. \tag{2.6}$$

But how to formulate the counteroffer? The problem becomes more tractable if instead of the total bid price u_i, the sellers know the value that order i places on slot j. This information can be represented as the *value matrix* P, where

$$p_{ij} = \text{value to order } i \text{ of slot } j, \tag{2.7}$$

in which case

$$x_i = \sum_j c_{ij} h(p_{ij}),$$

where h is some function of the p_{ij}, to be modeled in later sections. With this formulation of the objective function, the sellers' revenue can be maximized using various techniques, as discussed in later sections.

Of course, the bidders do not disclose information regarding P to the seller. Therefore, our analysis will depend largely on the assumption that the matrix P can be somehow inferred from the bid data using facts about the viewership tensor V. Various techniques for modeling P will be discussed in the next section.

Section 3: Modeling the Value Matrix

The first step in solving our optimization problem is creating an algorithm that will infer the value matrix P given the bid prices and the viewership tensor. In the test case, we know V since data is historical. In the real-world scenario, V for future slots can be estimated from historical viewing patterns. (For instance, slots at the same time each week would have similar viewing patterns.)

Impressions-Based Approaches

We first use several kinds of *impressions-based approaches*. In these methods, we infer the value matrix by using demographic data.

Total Impressions. The simplest way to model P is to base it on the total number of impressions for each desired slot:

$$p_{ij} = (\text{total bid } i)(\text{impressions for slot } j)\left(\frac{\text{value to } i}{\text{desired impression}}\right) = u_i\left(\sum_{a,g} v_{jag}\right) w_i, \tag{3.1a}$$

where w_i, the *weighted value per impression*, is defined by

$$w_i = \left(\sum_{j,a,g} b_{ij} v_{jag}\right)^{-1}. \tag{3.1b}$$

Note that w_i contains impressions only for the desired slots, since those are the only ones that are valued in the bid u_i. Moreover, if we define

$$p^*_{ij} = b_{ij} p_{ij}, \tag{3.2a}$$

then $p^*_{ij} = 0$ if the buyer didn't want slot j. Working with p^*_{ij} instead of p_{ij} will ensure that a buyer will never be awarded a slot (s)he didn't request. With this definition, we have the following constraints:

$$\sum_j p^*_{ij} = u_i, \qquad i = 1, \ldots, i_{\max}, \tag{3.2b}$$

which reflects the fact that the sum of the inferred values of the bid items must equal the bid price.

For an illustration, see Table 3.1. Here the buyer wants to buy slots A and B for \$272. The total number of impressions for those two slots is $16 (= 7 + 9)$, so $w_i = 1/16$. Hence the buyer values impressions at $\$17 (= \$272 w_i)$ each. Hence slot A has an intrinsic value of

	Impressions					Values					
	Demographic					Total	Demographic				Demographic
Slot	(1,1)	(1,2)	(2,1)	(2,2)	Total	Model	(1,1)	(1,2)	(2,1)	(2,2)	Model
A	*1*	*2*	*1*	*3*	*7*	119	0.04	0.15	0.04	0.22	124
B	*2*	*3*	*2*	*2*	*9*	153	0.09	0.22	0.09	0.15	148
C	3	3	1	1	8	136	0.13	0.22	0.04	0.07	128
Desired	*3*	*5*	*3*	*5*	*16*	Sum of Squares:		68			

Table 3.1. Inferring P values from a total bid $u_i = 272$. Here there are just four different demographics, rather than 30. Italicized entries: slots desired by bidder.

\$119($= 17 \cdot 7$), B has an intrinsic value of \$153, and slot C has an intrinsic value of \$136, as shown in the "total model" column.

We note that if $p_{ij} < r_j$, it may be desirable to set this bid equal to zero so there is never a chance the slot will be assigned to buyer i. This will be useful when we start trying to assign slots individually, rather than as part of a bid. In that case, we declare any bid where $p_{ij} < r_j$ to be *infeasible*, and we define a new *feasible value matrix* Q given by

$$q_{ij} = p_{ij} H(p_{ij} - r_j), \tag{3.3}$$

where $H(\cdot)$ is the Heaviside step function. For our data, this case actually happened quite often:

- 85% of the p_{ij} came in under the reserve price.
- If we restricted our analysis to p^*_{ij}, the rate dropped to 46%.
- Though 61 out of 300 complete bids would be rejected because they did not satisfy (2.3), no bidder had all values of $p_{ij} < r_j$. Hence a buyer whose complete bid would be summarily rejected may still be able to compete for certain slots within that bid.

One key test is whether these estimates are stable to perturbations. Therefore, we took the test data, ran the algorithm, and found the P matrix. Then we perturbed the bid amounts for the test data as follows:

$$u_i \mapsto u_i(1 + X), \qquad X \sim \mathcal{N}(0, (0.05u_i)^2). \tag{3.4}$$

In other words, we added a normally distributed variable with standard deviation equal to 5% of the original value. When we did this, the maximum relative error in p_{ij} was 15.1%. This is to be expected, since from (3.1a) we have that p_{ij} is linear in u_i, so the sensitivity

$$\frac{\partial p_{ij}}{\partial u_i}$$

is simply a constant depending on the entries in B and V.

Demographic-Based Approach. A more complicated (and realistic) impressions-based approach is given by valuing impressions from different demographics differently. In that case, we have

$$\begin{aligned} p_{ij} &= (\text{total bid } i) \sum_{a,g} \left[\begin{matrix} \text{impressions for slot } j \text{ and} \\ \text{demographic } (a,g) \end{matrix} \right] \left[\frac{\text{value to } i}{\text{desired impression in } (a,g)} \right] \\ &= u_i \sum_{a,g} v_{jag} w_{iag}, \end{aligned} \tag{3.5a}$$

where w_{iag} is the *weighted value per demographic*. Here this is defined by noting that if a demographic has a higher number of impressions, it is assumed more valuable. We assume a linear relationship; hence

$$w_{iag} \propto \sum_j b_{ij} v_{jag},$$

where the right-hand side is just the total number of impressions in demographic (a, g) in the bid. The normalization factor α_i is found by solving (3.2):

$$\begin{aligned} u_i &= \sum_j b_{ij} p_{ij} = u_i \alpha_i \sum_j b_{ij} \left[\sum_{a,g} v_{jag} \left(\sum_j b_{ij} v_{jag} \right) \right] \\ \frac{1}{\alpha_i} &= \sum_{a,g} \left(\sum_j b_{ij} v_{jag} \right)^2 \\ w_{iag} &= \sum_j b_{ij} v_{jag} \left[\sum_{a,g} \left(\sum_j b_{ij} v_{jag} \right)^2 \right]^{-1}. \end{aligned} \tag{3.5b}$$

Upon combining (3.5), we see that the value of a demographic varies quadratically with the number of impressions, reflecting the weighting factor.

For an illustration, see Table 3.1. In this case, most of the impressions are from people in demographics (1,2) and (2,2). Therefore, these impressions are assumed more desirable, and hence receive a higher weighting. In particular, we have

$$w_{i11} = w_{i21} = \frac{3}{68}, \qquad w_{i21} = w_{i22} = \frac{5}{68}.$$

The values associated with each slot are listed in the far right column. Note that the values for slots A and B sum to \$272, as they should.

In this case, our results showed similar behavior under the noise in (3.4). In particular, the maximum relative error in p_{ij} was 16.9%. Again, this is to be expected, since from (3.5a) p_{ij} remains linear in u_i.

In addition, the value matrix compared this way:

- 88% of the p_{ij} came in under the reserve price.
- If we restricted our analysis to p^*_{ij}, the rate dropped to 45%.
- None of the 61 complete bids that would be rejected would be rejected piecemeal, just as in the previous case.

There are several drawbacks to our demographic-based approach. First, it implicitly assumes that value is given *solely* by demographics. However, slot value may also arise from other issues (time of campaign start, clustering of spots, etc.). Thus our estimated p_{ij} may be quite different from the buyers' values of p_{ij}. This may lead to counteroffers

	Original Slot			Revised Slot		
Buyer	A	B	Total	A	B	Total
1	1500	500	2000	2250	750	3000
2	1000	750	1750			

Table 3.2. Unusual behavior needed to guarantee slot.

that have inferior subsets of slots, or prices not as heavily discounted as would lead to successful settlement.

In addition, the combination of this valuation approach with our assignment algorithm may lead to perverse results. Consider the situation outlined in Table 3.2. With the original bids, buyer 1 will receive slot A only because his bid for B fell $250 short. The sellers could respond that the buyer needs to up his bid a certain amount to be competitive for slot B. (This kind of communication can occur in asynchronous auctions.)

But note that given the value system, A is three times more valuable to the buyer than B. Hence the bid would have to be increased by $1,000 (four times the differential) in order for the bid on B to be competitive, which implies an overbid of $750 in the value of slot A.

Market-Based Approach

A different algorithm is given under a *market-based approach*. In this approach, we try to deduce p_{ij} from the bids alone. For instance, as a first model we could assume that buyer i assigns an intrinsic value p_{ij} to each slot, if it were bought alone. However, the value of subsequent slots is reduced by a *time discount factor* δ_{ij}. Hence the equation of each bid is given by

$$\sum_j \delta_{ij} p_{ij} = u_i, \qquad i = 1, 2, \ldots, i_{\max}, \tag{3.6}$$

which replaces (3.2b) in our analysis. The δ_{ij} are assumed known, and can take two forms:

- Since each additional viewing usually produces fewer *unique* impressions, we say that δ_{ij} decreases with each additional showing:

$$\delta_{ij} = (\delta_i)^{k_j}, \qquad k_j = \sum_{m=1}^{j-1} b_{im}. \tag{3.7a}$$

  In other words, k_j is the number of showings before slot j.
- Bidders may wish to cluster their commercials together. In that case, δ_{ij} decreases with the time from first showing:

$$\delta_{ij} = (\delta_i)^{t_j - t_{j_*}}, \qquad j_* = \min_j \{b_{ij} = 1\}. \tag{3.7b}$$

  where t_j is the time of slot j. Here j_* is just the index of the first slot bid. **IMPORTANT:** The test data is not currently ordered chronologically; it would have to be sorted first if this algorithm to be tested.

Section 4: Maximizing Slot Values Without Replacements

We begin by considering the case of counteroffers *without replacements*; that is, if a buyer loses out on a particular slot to another buyer, the buyer receives fewer slots, rather than the same number of slots with different assignments. Since we're not going to make replacements, we don't want a bidder to be awarded a slot that (s)he did not initially want. Therefore, we work with only the starred quantities.

Feasible Values

We begin by considering the case of feasible values. Hence

$$x_i = \sum_j q^*_{ij} c_{ij}, \tag{4.1a}$$

where q^*_{ij} is defined similarly to p^*_{ij}. If $\pi_j = 1$, we may maximize f by assigning slot j to the person who bid the most for it, and not to anyone else:

$$c_{ij} = \begin{cases} 1, & i = \arg\max_i q^*_{ij}, \\ 0, & \text{else.} \end{cases} \tag{4.1b}$$

These counteroffer results were stable under noise. In the case where (3.1) was used to calculate p_{ij}:

- The relative change in the *number* of assignments was 1.4%.
- The relative change in f was 2.6%.
- However, the fraction of assignments that changed was 43.9%.

So the relatively small change in the counteroffer total was caused by switching a large number of assignments. This is consistent with the top two values of p_{ij} being close together for a certain j. Hence assignments were switched between bidders even as the p_{ij} changed by small amounts.

Similar results held when we used (3.5) to calculate p_{ij}:

- The relative change in the *number* of assignments was 1.2%.
- The relative change in the sum of the counteroffers was 1.2%.
- The fraction of assignments that changed was 50%.

The final revenue results using (3.5) are as robust to noise as those using (3.1). Since (3.5) provides a finer-grained approach to estimating P, **ALL FURTHER RESULTS WILL BE CALCULATED USING (3.5)**.

This procedure maximizes the sellers' revenue if every counteroffer is accepted, but in practice this does not happen. Therefore, it is useful to measure the *Hamming distance* d_i of each bid:

$$d_i = \sum_j |b_{ij} - c_{ij}|. \tag{4.2a}$$

This measures how close the slots in the counteroffer to bidder i matches the bidder's original choice of slots, and is thus a rough measure of how "happy" bidder i would be. Another useful statistic is the percentage of requested slots not filled for bidder i:

$$d_{i\mathrm{r}} = d_i \left(\sum_j b_{ij} \right)^{-1}, \tag{4.2b}$$

where the subscript "r" refers to "relative". We also compute β, the total percentage of slots filled:

$$\beta = \sum_{i,j} c_{ij} \left(\sum_j \pi_j \right)^{-1}. \tag{4.3}$$

To measure the effectiveness of the revenue side, we calculate two statistics. The first is the ratio of the revenue to the total reserve:

$$f_{\mathrm{r}} = f \left(\sum_j \pi_j r_j \right)^{-1}. \tag{4.4a}$$

Note that f_{r} need not be greater than equal to 1, since the bid structure ensures that many orders will be unfilled, even with replacements. Therefore, to get a better handle on how much revenue the sellers received based on the actual assignments, we calculate

$$f_{\mathrm{a}} = f \left(\sum_{i,j} c_{ij} r_j \right)^{-1}, \tag{4.4b}$$

where the subscript "a" refers to "assigned." Here the denominator is the total reserve for all assigned slots. Hence $f_{\mathrm{a}} > 1$ if only feasible bids are used.

Figure 4.1 shows a histogram of the percentage of requested slots filled for each bidder for our test data. Note this is just $1 - d_{i\mathrm{r}}$. We see that without replacements, very few of the bidders receive any substantial number of slots at all. However, we note from our discussion in §2 that many of the bids are not feasible; *i.e.*, p_{ij} is smaller than the reserve price.

In Figure 4.2, we adjust our calculation so that we remove any infeasible bids from our calculations. So in (4.2b) we sum over only those slots j for which $q_{ij} \neq 0$. As one can see, this affects only a few bidders—those who had slots filled anyway. Those bidders saw their percentage rise.

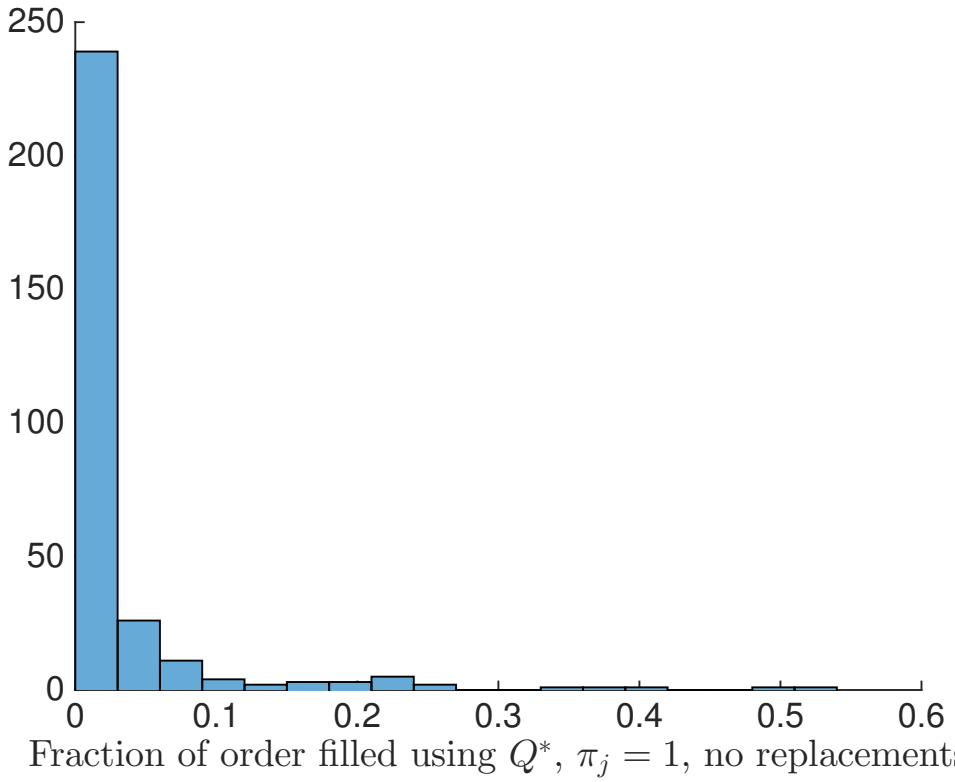


Figure 4.1. Distribution of percentage of requested slots accepted $(1 - d_{i\mathrm{r}})$, $\pi_j = 1$.

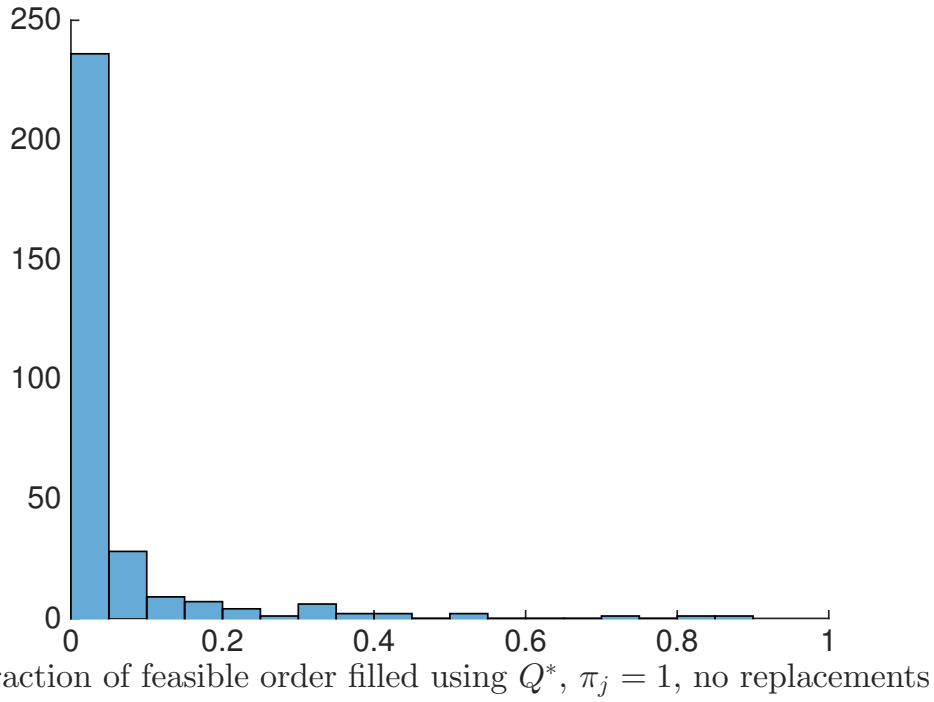


Figure 4.2. Distribution of percentage of feasible requested pods accepted, $\pi_j = 1$.

Table 4.1 shows the values of f_{r}, f_{a}, and β for this case. The fact that so many bidders had low fill rates is reflected in the fact that β is only 0.14, which explains the low value

Table 4.1. Statistics for different options of the maximum bid procedure.

	f_{r}		f_{a}		β	
	$\pi_j = 1$	$\pi_j = 4$	$\pi_j = 1$	$\pi_j = 4$	$\pi_j = 1$	$\pi_j = 4$
Q^*	0.19	0.16	2.16	2.12	0.14	0.12
P^*	0.31	0.26	0.89	0.89	0.30	0.25

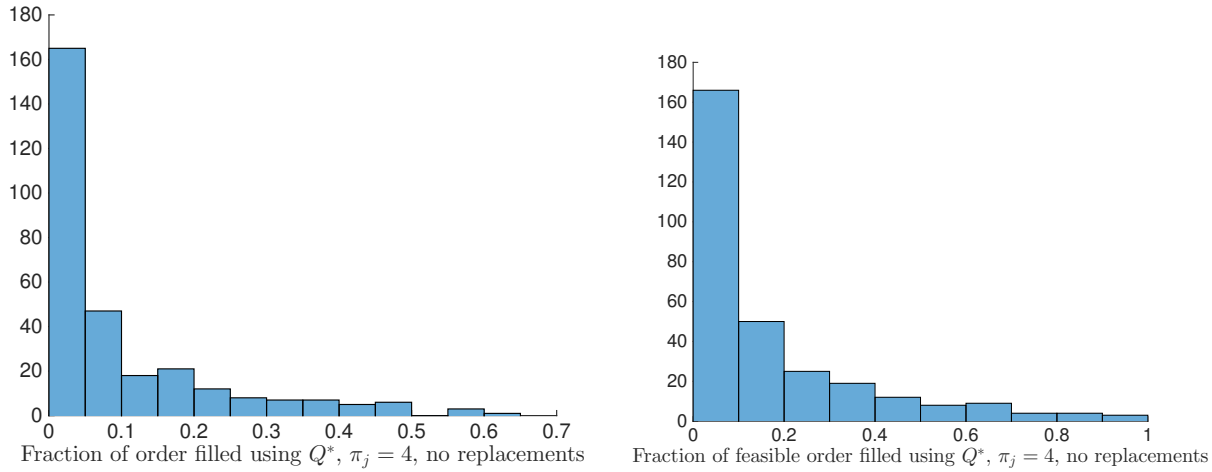


Figure 4.3. Distribution of requested pods accepted, π_j=4.

of $f_{\mathrm{r}} = 0.19$. However, $f_{\mathrm{a}} = 2.16$, reflecting the fact that bids were accepted only if they exceeded the reserve price, and on average the bids were over twice the reserve price.

Another case that can be simply analyzed is that of multiple pods for each slot. Rather than assuming that only one commercial can be placed in each slot, we now assume that π_j commercials can be placed in slot j. In that case, (4.1b) is replaced with the following:

$$c_{ij} = \begin{cases} 1, & i \in M_j, \\ 0, & \text{else}, \end{cases} \qquad M_j = \{\text{indices } i \text{ of } \pi_j \text{ largest } q^*_{ij}\}. \tag{4.5}$$

Note that (4.5) reduces to (4.1b) if $\pi_j = 1$ for all j. Note also that this formulation is flexible enough to handle cases where the slots have differing lengths.

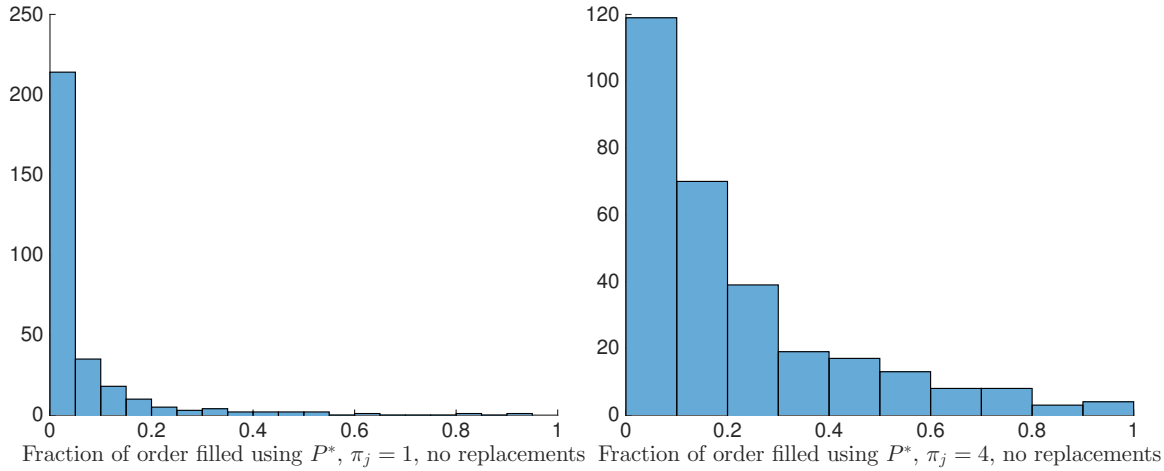
We expect that as π_j increases, there will be fewer rejected bids, and d_i should decrease. These results are shown in Figure 4.3. Though d_{i} did increase, β decreased, as reflected in the values in Table 4.1. This occurs since with more pods per slot, there may no longer be enough bids to fill every pod.

It may seem somewhat counterintuitive that f_{r} decreased as well. But remember that f_{r} is the revenue *relative* to the maximum capacity. In this case, the absolute revenue increased nearly fourfold, but with four pods per slot, the total reserve *over all pods* increased by a factor of four as well. Similarly, f_{a} declined somewhat because the additional assigned bids were valued less than the winning bids with $\pi_j = 1$. Given these somewhat counterintuitive results, it is useful to examine the total absolute revenue as well when evaluating counteroffers.

In addition, there are still large numbers of bidders with very few slots awarded, which is keeping $f_{\mathrm{r}} < 1$. This is why replacements are common practice.

All Wanted Values

Given the low value of f_{r}, we now consider retaining those values for which $p_{ij} < r_j$. Though the amount gained from the counteroffer on each slot may be less than the reserve price, if the slot goes unfilled for lack of feasible bids, then the seller receives no payment at all.



Fraction of order filled using P^*, $\pi_j = 1$, no replacements Fraction of order filled using P^*, $\pi_j = 4$, no replacements

Figure 4.4. Distribution of requested pods accepted.

Therefore, in the above analysis we replace q^*_{ij} with p^*_{ij}. The results are shown in Fig. 4.4. In this case we don't have to do a separate computation for the feasible bids.

For both values of π_j, we see improvement in the allocation of bids (the left of Fig. 4.4 should be compared with Figs. 4.1 and 4.2; the right of Fig. 4.4 should be compared with Fig. 4.3). This makes sense, since now even infeasible bids can be allocated. This improvement is reflected in Table 4.1, as the values for f_{r} and β are larger than their Q^* counterparts for both values of π_j. However, the value of f_{a} is less, reflecting the fact that the additional bids allotted have values that are smaller than the reserve price, which drags f_{a} down.

Section 5: Other Counteroffers Without Replacements

Using Total Bids

Another approach to maximizing the revenue is to look at the total bids for each bidder, rather than the value of each slot. One example is the "max-sum" algorithm, which holds only for $\pi_j = 1$:

1. Set $C = O$.
2. Find
$$\max_i \sum_j b_{ij} p^*_{ij} \text{ or } \max_i \sum_j b_{ij} q^*_{ij}.$$
Let I be the index of the maximum.
3. Give the winning bidder all his/her bids by letting $c_{Ij} = b_{Ij}$ for all j.
4. For all j such that $b_{Ij} \neq 0$, set $b_{ij} = 0$, reflecting the fact that the slot has been awarded.
5. Set $b_{Ij} = 0$ for all j to remove bidder I from further competition.
6. Loop through steps #2–#5 until $B = O$.

For similar values of π_j and choice of P^* or Q^*, we would expect this algorithm to lead to lower values of f_{r}, since one doesn't have the flexibility to pick the maximum price for each slot. More details of this algorithm may be found in the chapter by Emerick and Goldwyn.

Another option is to replace #2 with

2a. Find
$$\max_i \sum_j b_{ij} p^*_{ij} \left(\sum_j b_{ij} \right)^{-1} \text{ or } \max_i \sum_j b_{ij} q^*_{ij} \left(\sum_j b_{ij} \right)^{-1}.$$

This just corresponds to looking at the largest bid amount per slot bid. This may reduce revenue even further. To see why, consider a large bid spread over many slots. This bid would be prioritized in the maximum bid algorithm, but with a small average, would be rejected in this one.

Using the Hamming Distance

Another way to perform the optimization on f is to take the Hamming distance into account.

For instance, we could maximize f given the constraint that all orders must be filled *in toto*, so
$$d_i = 0 \text{ (order filled) or } d_i = \sum_j b_{ij} \text{ (order rejected).}$$

In practice, this means very few orders would be filled (in our test case, only two of 300 orders were filled with $\pi_j = 1$, and nine of 300 orders were filled with $\pi_j = 4$).

However, this algorithm could be used as a preprocessing step. In particular, once the complete orders are filled, zero out the corresponding rows i and columns j in P and Q. Then use the standard algorithm as described in §4 to fill the remaining orders as best as possible. This may not yield a particularly optimal f in the case of no replacements, since if there is substantial overlap, there may be few slots left which the remaining bidders desire.

More details about using the Hamming distance in this way can be found in the chapter by Narayanan.

Alternatively, we could use the Hamming distance in conjunction with the maximum bid algorithm from §4 by replacing #2 with

2b. Run the maximum bid algorithm to get a provisional assignment matrix. Find the bid with

$$\min_i d_i.$$

This just corresponds to selecting the bidder that the seller can make happiest at each iteration. This gives an f_{r} value roughly the same as for #2a.

Other possibilities include:

- We might want to minimize the number of changes made to bids:

$$\min \sum_i d_i. \tag{5.1}$$

- If we have a hierarchy of "best" customers, we could introduce weights to the sum in (5.1) to ensure that the best customers get their orders satisfied.
- We also might want to separate the orders into sets. If d_i is close to 0 for some buyers, we may wish to reduce d_i for those buyers further at the expense of other buyers for whom d_i is already large. In essence, given that a buyer is not going to get much of what she bid for and will be unhappy anyway, why not take some additional slots from her to make another reasonably happy buyer even happier?

Section 6: Simple Counteroffers With Replacements

We now consider the case where replacements are made. In that case, we would work with the full matrices P and Q, since we need to know the inferred values for the slots the buyer didn't bid on. So (4.1a) would hold with the starred variables replaced with the unstarred ones.

We begin with a simple optimization procedure:

1. Set $C = O$.
2. Find
$$\max_{i,j} p_{ij}.$$
Let I and J be the indices of the maximum.
3. Assign slot J to the winning bidder by setting $c_{IJ} = 1$.
4. Update $x_I \to x_I + p_{IJ}$, reflecting the fact that the bid has been awarded and must be paid for.
5. Set $p_{iJ} = 0$ for all i to reflect the fact that the slot has been assigned.
6. If $p_{Ij} + x_I > u_I$, set $p_{Ij} = 0$ for all j to reflect the fact that adding this slot to bid I would cause the counteroffer to exceed the original bid. (This is equivalent to the constraint that
$$x_i \le u_i \tag{6.1}$$
for all i.)
7. (Optional) If
$$\sum_j c_{Ij} = \sum_j b_{Ij}, \tag{6.2}$$
set $p_{Ij} = 0$ for all j to keep the number of slots assigned to bidder I less than or equal to the number of slots bid. If we include this step, we call the replacements *constrained*.
8. Loop through steps #2–#7 until $P = O$.

Note that if we didn't impose the constraint in (6.1), the algorithm would assign any slot for which $p_{ij} > 0$ for some i, thereby potentially exceeding the budget u_i. However, step #7 need not be implemented, since one could imagine giving an advertiser two "cheaper" slots in place of one expensive slot that was given to another bidder. However, if this process is repeated many times, the commercial could saturate, which may be unwanted. Hence we will perform the algorithm both with and without this constraint.

In the case of replacements, d_i measures both unfulfilled slots and replaced slots. In addition, $d_{i\mathrm{r}}$ is no longer bounded above by 1. In particular, if the number of counteroffer slots equals the number of bid slots, but every slot has been replaced by another, $d_{i\mathrm{r}} = 2$.

Therefore, we define the following more useful statistic:

$$\gamma_i = \left(\sum_j b_{ij} c_{ij} \right) \left(\sum_j b_{ij} + c_{ij} - b_{ij} c_{ij} \right)^{-1}. \tag{6.3}$$

Here the numerator is the number of slots the bidder bid on and received. The denominator is the number of slots the bidder bid plus the number received, minus the number bid on and received. Hence the denominator is the number of slots the bidder bid on *or* received. Note that γ_i runs from 0 (bid and received totally distinct) to 1 (bid and received the same).

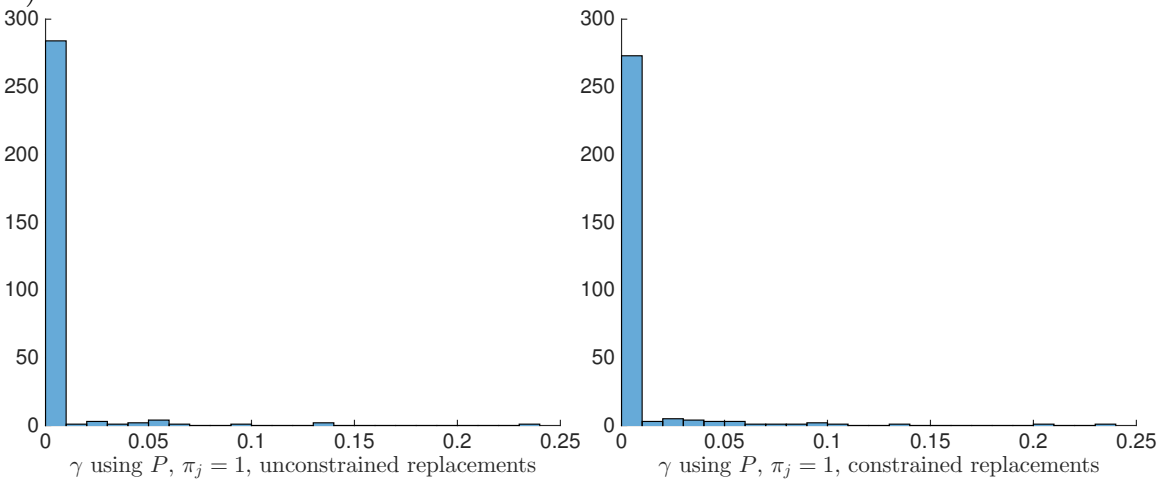


Figure 6.1. γ for $\pi_j = 1$ using P. Left: unconstrained. Right: constrained.

Figure 6.1 shows the results for γ_i for $\pi_j = 1$ using P for both the constrained and unconstrained case. Note that most of the γ_i are near 0, which reflects that we are using P (all possible bids), rather than the sparser matrix Q (all feasible bids). Also note that the γ_i distribution is more skewed to the left in the unconstrained case. This suggests that more unwanted slots can be given to each bidder and still remain under his budget. With more unwanted slots, the denominator of (6.3) increases, driving γ_i down.

Table 6.1. Statistics for different replacement options.

	f_r		f_a		β	
	$\pi_j = 1$	$\pi_j = 4$	$\pi_j = 1$	$\pi_j = 4$	$\pi_j = 1$	$\pi_j = 4$
P (unconstrained)	0.47	0.43	0.47	0.43	1.00	1.00
P (constrained)	0.46	0.41	0.46	0.41	1.00	1.00
Q (unconstrained)	0.26	0.22	2.14	2.10	0.24	0.22
Q (constrained)	0.26	0.22	2.14	2.11	0.23	0.21

These results are quantified in Table 6.1. First, note that to two decimal places, all the slots are filled ($\beta = 1$). This is because we are using P, so all slots are available for placement, and we can do replacements. We note this is true for both the constrained and

unconstrained cases. This reflects the fact that

$$\sum_{i,j} b_{ij} > j_{\max}. \tag{6.4a}$$

(For our data, the left-hand side of (6.4a) is 69,292.) Hence even if the algorithm constrains the number of slots each bidder receives, one can still assign all the slots.

These additional assignments do drive up revenue, even if the slots are given away for less than the reserve price; note that the values of f_{r} are roughly 50% higher than their counterparts in Table 4.1. Also, since almost all the slots were assigned a bidder, the denominators in (4.4) are nearly identical. Hence f_{r} and f_{a} have the same values. We note little change between the constrained and unconstrained case. This is due to the fact that if all the slots are assigned in the constrained case, lifting the constraint isn't going to change the overall assignments (and hence the revenue) by very much.

We may also relax the (implicit) assumption in the above that $\pi_j = 1$. We may do this by replacing #5 with

5a. If

$$\sum_i c_{iJ} = \pi_J,$$

set $p_{iJ} = 0$ for all i to reflect the fact that all pods for the slot have been assigned. Note this reduces to #5 in the case of $\pi_j = 1$.

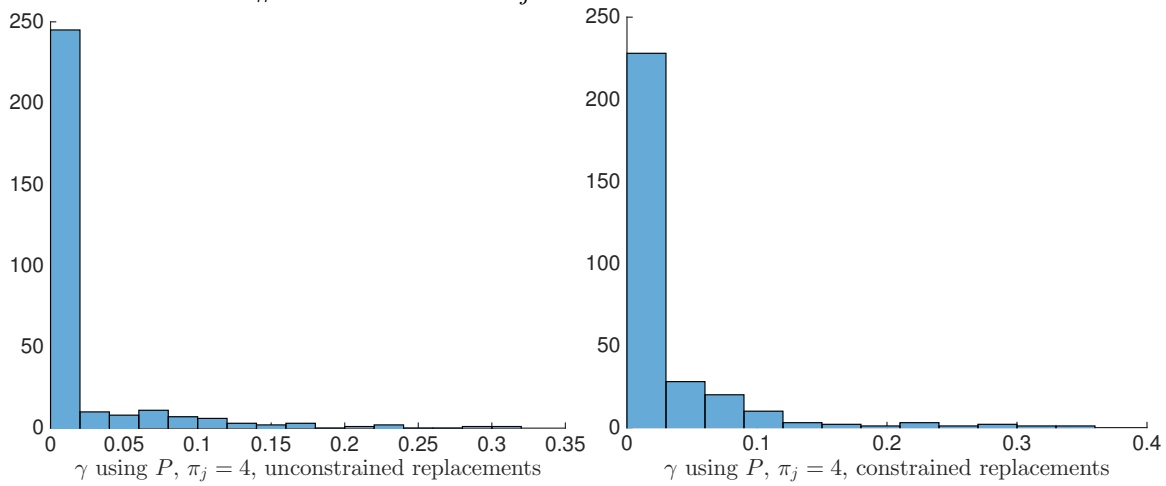


Figure 6.2. γ for $\pi_j = 4$ using P. Left: unconstrained. Right: constrained.

The results are shown in 6.2. As expected, the distributions for γ_i shift to the right, reflecting the increased flexibility available in assigning slots. β still remains near 1, which is indicative of the following inequality replacing (6.4a):

$$\sum_{i,j} b_{ij} > \sum_j \pi_j, \tag{6.4b}$$

which is also satisfied by our test data. In other words, there are so many bid slots that even with four times as many pods, we can still fill them all. Hence f_{r} and f_{a} are the same.

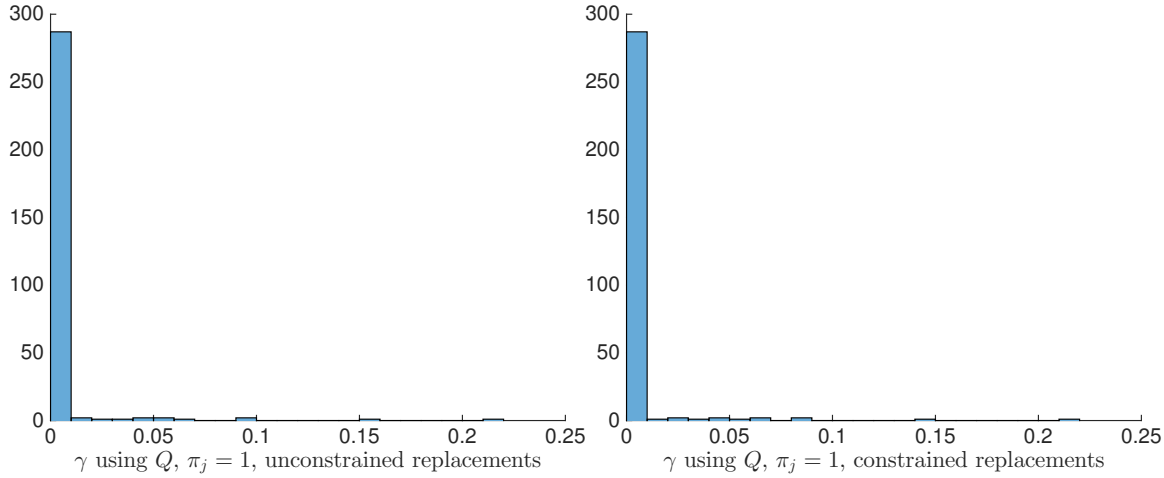


Figure 6.3. γ for $\pi_j = 1$ using Q. Left: not using step #7 (unconstrained). Right: using step #7 (constrained).

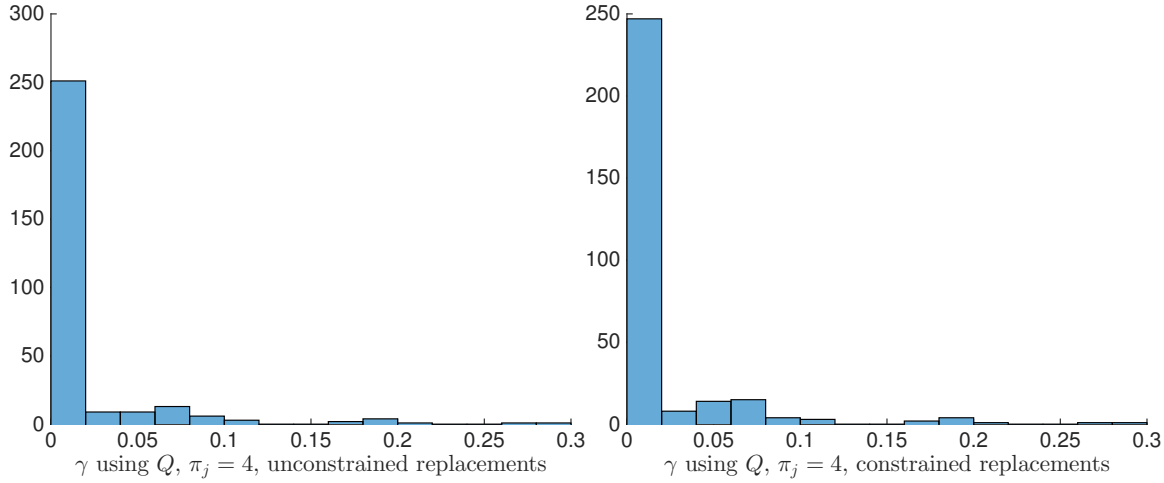


Figure 6.4. γ for $\pi_j = 4$ using Q. Left: not using step #7 (unconstrained). Right: using step #7 (constrained).

Again, the maximum revenue goes up by a factor of 4, but the sellers are adding 4 times as many slots at less than the reserve price, so f_{r} declines slightly. Again, the values are roughly 50% higher than their counterparts in Table 4.1.

We may also replace P by Q in our algorithm; these results are shown in Figs. 6.3 and 6.4. (Note we use all bids—not just feasible ones—for our calculation of γ_i, since the buyers don't know which bids are feasible.) The results for γ_i are not appreciably different from those for P.

As expected, the values for β are much less (since the population of feasible bids is very small). However, they remain about twice as large as their counterparts in Table 4.1. The figures for f_{r} remain about 50% larger than the case without replacements. However, the values of f_{a} are roughly similar. Given the feasibility constraint, the sellers are guaranteed to replace slots with others with similar value. Hence the ratio f_{a} of assigned revenue should remain roughly the same.

Section 7: Counteroffer Complications

We now wish to discuss the following issues, any one of which would cause the optimization problem to become nontrivial, necessitating greedy algorithms or other approaches.

Discounts

In general, a complete bid should command a premium compared to getting individual slots. Hence the definition of x_i in (4.1a) is too simplistic. A more realistic definition is

$$x_i = \sum_j p_{ij} c_{ij} (1 - \rho_{ij}), \tag{7.1}$$

where ρ_{ij} is the *discount factor*. (Here p_{ij} can be replaced with q_{ij} or starred, depending on the algorithm.)

There are several different types of discounts:

- a fixed discount $\rho_{ij} = \rho \operatorname{sgn}(d_i)$ for any customer whose order was not completely fulfilled.
- a discount that varies by customer, but just reflects complete fulfillment: $\rho_{ij} = \rho_i \operatorname{sgn}(d_i)$. (This reflects the fact that better customers might get larger discounts.)
- a discount factor that increases with d_i, so (in the simplest linear case) $\rho_{ij} = \rho d_i$ (fixed discount) or $\rho_{ij} = \rho_i d_i$ (preferred customer discount).
- a discount factor that depends on the individual replacements made. For instance, a higher discount could be offered
  - — if the first requested slot is replaced (it may be important to launch a project on a certain day or time),
  - — if two slots requested to be closely placed are replaced with ones spaced further apart (it may be important to reinforce an impression over a short period of time),
  - — other discounts based upon δ_{ij} from §3,
  - — etc., etc.

In this case the counteroffer matrix C is given by

$$c_{ij} = \begin{cases} 1, & j = \arg\max\limits_i p_{ij}(1 - \rho_{ij}), \\ 0, & \text{else}, \end{cases} \tag{7.2}$$

or an equivalent form based on (4.5). These problems are much more difficult to solve. In particular, this cannot be handled by straightforward optimization, and will require greedy algorithms or other techniques to manage the tradeoffs when assigning a slot to one order over another.

Buyer	Values Slot A	B	C	Counteroffer Slot B Awarded to 2	1
1	*30*	*10*	1	27	40
2	1	*11*	*10*	21	9

Table 7.1. Tradeoffs with 10% discount. Italicized entries are desired slots.

This problem is illustrated in Table 7.1. Buyer 1 wants slots A and B, and buyer 2 wants slots B and C. A 10% discount is given if a buyer doesn't receive her request. If we follow the algorithm described by (4.1b), the payoffs are as described in the "2" column, with a total counteroffer of \$48. But if instead we give slot B to buyer 1, the loss in revenue from slot B is more than made up for by the reduced amount of the discount given to buyer 2, and the total counteroffer is more: \$49.

Settlement Price

When considering the actual counteroffer value x_i to make, the seller must act strategically. If one sets x_i too close to u_i for an inferior assignment, the buyer may balk, in which case the seller receives 0. Set x_i too low, and the buyer will accept, but the seller may not be maximizing revenue. In between, there may be some negotiation where the settlement price s_i is less than x_i, but more than the seller would receive at either extreme.

A graph of possible relationships is shown in Figure 7.1. The dotted line $s_i = x_i$ indicates that the counteroffer was accepted. The thin line indicates a buyer who negotiates aggressively whenever $x_i > 1$ and walks away for $x_i > 2$. The thick line indicates a buyer who negotiates a little at first, then more aggressively as x_i grows, finally walking away for $x_i > 3$.

Obviously we want to keep x_i as close to the peak as possible. Tuning this would require knowing something about bidder i (and his/her negotiating history). One way this could be incorporated into the above model is using it to determine ρ_{ij} for a particular buyer.

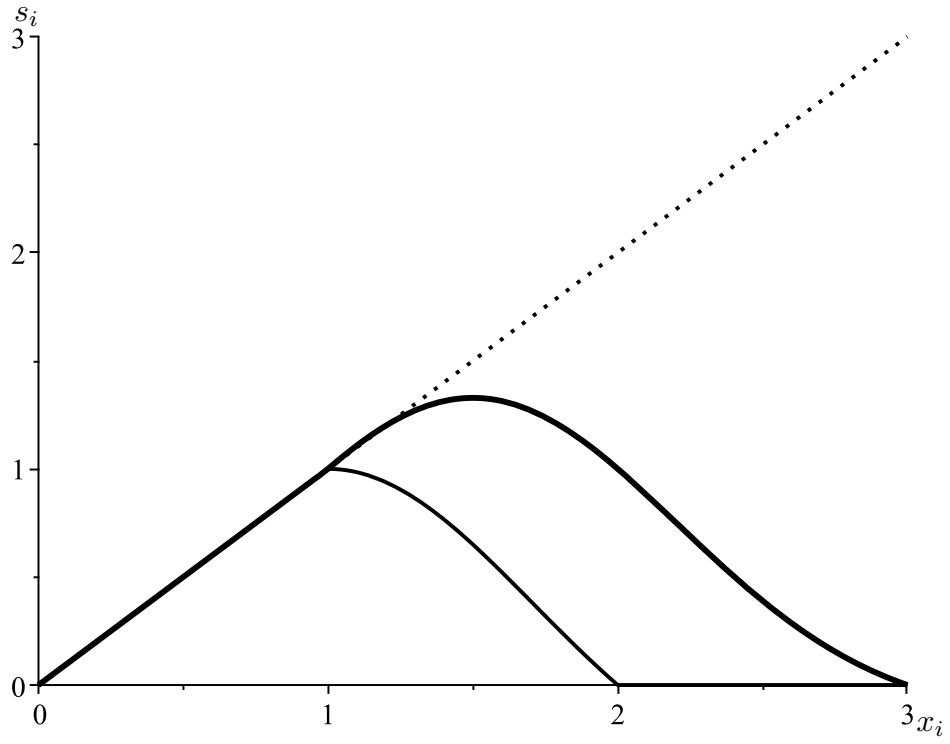


Figure 7.1. Settlement price s_i *vs.* counteroffer price x_i.
Dotted line: counteroffer is accepted.

Section 8: Conclusions and Further Research

In order to maximize the revenue generated by commercial slots during television programs, sellers run auctions where buyers can bid on particular slots. Since these bids may overlap on many slots, an algorithm must be designed to award winning bids and generate counteroffers to losing bids. In this manuscript, we analyzed several ways of running such auctions, and listed many other possibilities which can be explored in later work.

Unfortunately, the buyers bid on a combination of slots, rather than each slot individually. This makes it difficult for sellers to construct an effective counteroffer on a subset of the desired slots. Therefore, it is useful to infer the value each bidder places on each slot. Typically, buyers want to reach the largest number of viewers in some target audience based upon demographic and income data.

This motivated several ways to determine the value matrix P. The first is simply to use total impressions per slot (determined from historical data), and weight each slot accordingly, as in (3.1). The second is to look for common demographic characteristics among the requested slots, and assume that they are desired. Then one can weight each slot by the desired demographics, yielding the finer-grained approach (3.5) we used in most of the manuscript. Since both approaches are linear in the bid price u_i, the values are insensitive to noise in the bids.

Once the value matrix is determined, the slots can be assigned to the bidders. In the simplest case, there are no replacement slots awarded. Hence the counteroffers will largely be for partial bids. The simplest approach is merely to award each slot individually to the highest bidder. Depending upon financial considerations, it may be useful to restrict bids to only those which exceed the reserve price for a slot; this necessitates using the matrix Q defined in (3.3) instead of P.

We analyzed this algorithm using P and Q for various (constant) values of the number π_j of pods per slot, though our model can handle values of π_j which vary with j. We evaluated our results based upon the histogram of slots awarded per bidder, as well as the statistics β (slots filled), f_{r} (revenue as a fraction of reserve), and f_{a} (revenue as a fraction of assigned reserve).

In general, we found that for fixed pod number, β and f_{r} are maximized using P. This makes sense, since using P means the seller considers all bids, no matter how small. However, to maximize f_{a}, the seller should use Q, since in that case any assigned slot will be filled for more than the reserve price.

There are other counteroffer strategies that do not use replacements. One can assign slots based on the maximum total bid, rather than the total for each slot. This is discussed further in the chapter by Emerick and Goldwyn. One can also assign slots by maximizing the number of complete bids awarded; this is discussed further in the chapter by Narayanan.

Since the bid overlap is so great, typically sellers offer replacement slots to bidders so that the counteroffer is not just a few slots. We analyzed in detail one replacement algorithm, again based upon awarding slots to the maximum bidder. Since replacements

have associated costs, the algorithm ensures that bidders do not pay more than their original bids for the counteroffer with replacements. An additional optional constraint was implemented that ensures that the number of slots run doesn't exceed the original number bid.

When running the algorithm with all bids using P, the revised bids filled all the slots, no matter how many pods we had. This is consistent with the demand exceeding the supply. The revenue measurements were higher than in the case without replacements, and similar values occurred whether the sellers constrain the optimization using (6.2) or not. When running the replacement algorithm with Q, much fewer slots were filled, since the sellers accept only feasible bids. From the revenue side, whether it is better to use P or Q depends on whether one measures revenue using f_{r} or f_{a}.

Given the short time frame of the workshop, we have analyzed only the simplest cases. However, these results should illuminate the advantages and drawbacks of the few algorithms analyzed, and the techniques used can guide study of more complicated algorithms, as discussed below.

Further Research

There are other ways to deduce the value matrix other than the demographic-based approaches discussed above. In particular, one can determine the value matrix using data about the sequencing of the bid slots. There are also other ways to optimize the revenue besides just maximizing f. In particular, one may wish to maximize f_{r} or f_{a} instead. These changes would yield different algorithms.

In this manuscript, we assumed there was a "black box" algorithm that would transform the constraint bidders' wishes into bid orders, which would be incorporated with the other bid orders at the same time. However, it may be more advantageous to consider the types of orders sequentially. Once all of the bid orders have been assigned using the algorithms discussed here, then the constraint bids could then be accommodated with the "leftovers."

As discussed in §7, the seller may wish to award a discount to an unsuccessful bidder, beyond that which would accrue by not receiving the full complement of slots in the bid. These discounts can be implemented in many ways, and would complicate the optimization procedure for the revenue. In addition, the seller must also strategically price his counteroffer so as not to drive a bidder away for that auction.

In our work, we assumed that all bidders were rational, in that they were able to determine the value of slots themselves and bid appropriately. This assumption underlies the work we did in determining P. However, bidders may not be rational in this sense. In that case, the counteroffer prices we give may have little relationship to the value the bidder places on the subset of slots counteroffered.

On the other extreme, given any auction, there are ways for sophisticated bidders to game the system (for example, see the settlement price discussion in §7). Hence sellers would have to take that into account when choosing an algorithm.

Nomenclature

If a letter appears in upper and lower case, the upper case letter is a matrix or tensor, and the lower case letters are its entries. Equation numbers where a variable is first defined is listed, if appropriate.

a: indexing variable for age distributions (2.1).
B: bid matrix (2.2).
C: counteroffer assignment matrix (2.4).
d_i: Hamming distance for bid i (4.2a).
f: revenue (2.6).
g: indexing variable for gender (2.1).
$h(p_{ij})$: function relating x_i to p_{ij}.
I: index of winning bidder.
i: indexing variable for orders (2.2).
J: index of assigned slot.
j: indexing variable for slots (2.1).
k: exponent in time discount factor (3.7a).
M_j: set of acceptable bids for slot j in multiple pod case (4.5).
P: matrix of (inferred) values for each slot (2.7).
Q: matrix of (inferred) feasible values for each slot (3.3).
r_j: reserve price for slot j.
s_i: settlement price for bid i.
t_j: time of slot j (3.7b).
u_i: bid price for bidder i.
V: viewership tensor (2.1).
w: weighting factor, variously defined (3.1b).
X: normally distributed random variable (3.4).
x_i: counteroffer price to bidder i.
α_i: normalization factor.
β: percentage of slots unfilled (4.3).
γ_i: fill factor for replacement case (6.3).
δ: time discount factor (3.6).
π_j: pods in slot j (2.5).
ρ: bid discount factor (7.1).

Other Notation

a: as a subscript on f, refers to assigments (4.4b).

max: as a subscript on i or j, refers to the total number.
r: as a subscript, refers to relative change (4.2b).
$*$: as a subscript on j, refers to the first slot bid (3.7b); as a superscript on p or q, refers to values for unwanted slots being zeroed out (3.2a).

References

[1] M. Montes de Oca, *et al.*, "Prediction and optimal scheduling of advertisements in linear television," in *Proceedings of the 31st Annual Workshop on Mathematical Problems in Industry*, 2015.

No-Replacement Algorithms and Other Stuff

Brooks Emerick
Trinity College, Hartford CT 06106

Eli Goldwyn
Trinity College, Hartford CT 06106

June 24, 2016

The following is a summary of some of the things that Eli and I worked on throughout the week. I will summarize our contribution in the sections below, but the discussion provided here (especially nomenclature) should be combined and cross-referenced with Dr. Edwards' writeup. The basic outline is as follows: Section 1 contains a description of the matrices P^* and Q^* that are used to compute the winning bidder; Section 2 is a summary of the two leading algorithms (that do not consider counteroffers and replacements) used to distribute time schedules (orders) and the sensitivity of these algorithms to discounts; finally, Section 3 describes the number of time slot pods, denoted by π, and how this parameter affects the Hamming distance among other metrics.

[Note: in the discussion below, $i = 1, 2, \ldots, 300$ is the index ranging over the bidders; $j = 1, 2, \ldots n$ where n is the number of time slots; and $k = 1, 2, \ldots, 30$ is the index ranging over the demographics.]

1 The Value Matrix

To begin, we are given a vector of bids, $\boldsymbol{t} = [t_i]$, which (given the data) is a 300×1 column vector. Also, we are given two matrices, $B = [b_{ij}]$ and $V = [v_{jk}]$, where B is a large $300 \times n$ ($n \approx 14000$) matrix of ones and zeros, where n denotes the number of available time slots, and V is a matrix of size $n \times 30$ that contains the viewership numbers for each time slot over 30 demographic categories. The matrix B is a summary of the schedule for each bidder in the sense that $b_{ij} = 1$ if bidder i requests time slot j and $b_{ij} = 0$ otherwise. The matrix B is sparse as there are many time slots (~ 14000) but most bidders only bid on 230 time slots, on average. The matrix V is not too important but it allows us to generate the value matrix $P^* = [p^*_{ij}]$ through Erik's weighted average method. The construction of the matrix P^* is not trivial as it took us a full day to get the code to work. The m-file in the dropbox denoted

by `Value_Matrix.m` takes input values B, V, and $\boldsymbol{t}$ and outputs P^* (as `P_star`) along with other important matrices and vectors. The matrix P^* has the same sparsity pattern as B, but where B has a 1, P^* has a monetary value assigned to the time slot in column j. This value located in the $(i,j)^{th}$ spot of P^* is a proportion of bidder i's total bid for time slot j. The value is calculated using the demographic data from matrix V and represents both the monetary value of bidder i's bid on time slot j as well as the interest bidder i has in that specific demographic. Note that the following equation holds for all i:

$$t_i = \sum_{j=1}^{n} p^*_{ij}.$$

We also note here that P^* contains only the proportion of the full bid that bidder i has placed on time slot j.

Another matrix that we computed in order to maximize revenue is the matrix $Q^* = [q^*_{ij}]$. This matrix is a "sub-matrix" of P^* in the sense that Q^* is more sparse than P^*. To construct Q^*, we consider the $1 \times n$ row vector, $\boldsymbol{r} = [r_j]$, which contains the minimum bid value (reserve value) of each time slot. This vector is given and is denoted by `rawmin` in the data file. Essentially, for each i, we look at bidder i's bid value for time slot j in the matrix P^* and if it falls below the value r_j, then we zero out that value. This creates a somewhat more sparse matrix denoted by Q^*. That is,

$$q^*_{ij} = \begin{cases} p^*_{ij} & \text{if} \quad p^*_{ij} \geq r_j \\ 0 & \text{if} \quad p^*_{ij} < r_j \end{cases}, \quad \text{for all } i.$$

Now that we have defined these two matrices, we can move on. Please note that the notation for these matrices may be different from Dr. Edwards' notation. Also, we do not discuss the less sparse matrices P and Q here as the algorithms discussed below do not reference these matrices. We only reference P and Q if replacement bids and counteroffers are considered, which has only been done by Dr. Edwards while using the *Sum-Max* method discussed below.

2 The (No-Replacement) Algorithms

Throughout the week, we have worked to formulate an algorithm that generates a method for distributing time slots to the bidders that will maximize the seller's revenue. We will discuss the two methods below, but be sure to note that neither of these methods are sophisticated enough to provide a counteroffer to the bidders. The first method provides more revenue but sacrifices filling a requested time schedule completely. We will refer to this algorithm as the *Sum-Max* method. The second method provides less revenue overall but fills at least one order. This method is called the *Max-Sum* algorithm. Essentially, these algorithms boil down to summing the maximum of the columns of P^* or maximizing the row sum of P^*, respectively. Although the *Sum-Max* seems to (and most likely always will) maximize the revenue, we'll discuss how the *Max-Sum* method may actually work better if discounts were implemented into the system.

The *Sum-Max* Method

The *Sum-Max* method considers the columns of the matrix P^* and generates an assignment matrix, denoted by $C = [c_{ij}]$. Essentially, we pick the first time slot that has at least one bid for it and find the maximum p^*_{ij} bid value in that column and assign it to the corresponding bidder that made that particular bid. This is a very simple algorithm posed by Dr. Edwards that basically does the following:

[1] Construct a $300 \times n$ matrix of zeros and call it C.

[2] Choose time slot j (column j) of the matrix P^*.

[3] Find the bidder I such that p^*_{Ij} is a maximum, i.e., compute $p^*_{Ij} = \max_i\{p^*_{ij}\}$.

[4] Store a value of 1 in the I^{th} row and j^{th} column of the matrix C.

[5] Repeat steps [2] – [4] for all j.

This method formulates an assignment matrix C, which is a sub-matrix of B. Note that C has at most one 1 in each column and allocates each time slot to the highest bidder. To find each bidder's revenue and the total revenue, we compute the vector $\boldsymbol{x} = [x_i]$ and the value f, which are given by

$$x_i = \sum_{j=1}^{n} c_{ij} p^*_{ij} \qquad \Rightarrow \qquad f = \sum_{i=1}^{300} x_i.$$

The *Max-Sum* Method

The method that we considered today was motivated by filling at least one order so that we don't have to discount the counter-offer (discussed below). This method looks first at the global max bid and gives that bidder his entire schedule. Then, since we have given this time schedule away, we no longer consider these time slots and find the next bidder whose incomplete time schedule yields the maximum bid. All of this is computed from the matrix P^* in the following manner:

[1] Find the index I such that t_i is maximized. Store t_I as revenue.

[2] In the matrices P^* and B, zero out the row corresponding to I and zero out all columns that correspond to bidder I's time schedule.

[3] In the updated version of P^*, find the row I that maximizes the value $\sum_{j=1}^{n} p^*_{ij}$. Store this maximum value and add it to your revenue.

[4] Repeat steps [2] and [3] until the number of bidders is exhausted.

This method makes sure that the highest bidder is completely satisfied; however, it doesn't maximize the revenue for the seller. A good quality of this method is the fact that we could use different criteria for choosing the best bidder in each step. For example, instead of being greedy and picking the bidder that maximizes revenue at each step, we could potentially pick the bidder that has the highest bid per time splot or the minimum number of time slots remaining in order to fill their original order. These are different scenarios in which the algorithm could be used, but we see in the next section that they do not maximize revenue. We discuss the two methods in more depth below. We also note here that although the steps above do not include the generation of an assignment matrix C, the algorithm in MATLAB does create the matrix C.

Comparison

To compare the two methods, we compute the following metric:

$$f_r = f\left(\sum_{j=1}^{n} r_j\right)^{-1}.$$

This value can be interpreted as the ratio of our generated revenue (using the methods above) to the minimum revenue that would be obtained if we were given the minimum reserve value for each time slot. In a sense, this value is the proportion of the revenue we "could have" generated and so is a measure of how well our algorithm does in generating revenue. We find that $f_r \approx 0.3062$ when using the *Sum-Max* method and $f_r \approx 0.2755$ using the *Max-Sum* method. This indicates that the summing the column max is the superior method. If we run the same two algorithms using the matrix Q^* instead of P^*, then this method is still better. Using Q^* will theoretically increase the computational time since it is more sparse, but using this matrix basically means that we do not consider the cheap bidders – we're more picky in our selection. If this is the case, $f_r \approx 0.1936$ using the *Sum-Max* method and $f_r \approx 0.1673$ using the *Max-Sum*. These values make sense because Q^* has a lot more zeros than P^* so we actually accrue much less revenue when using Q^* to compare bids.

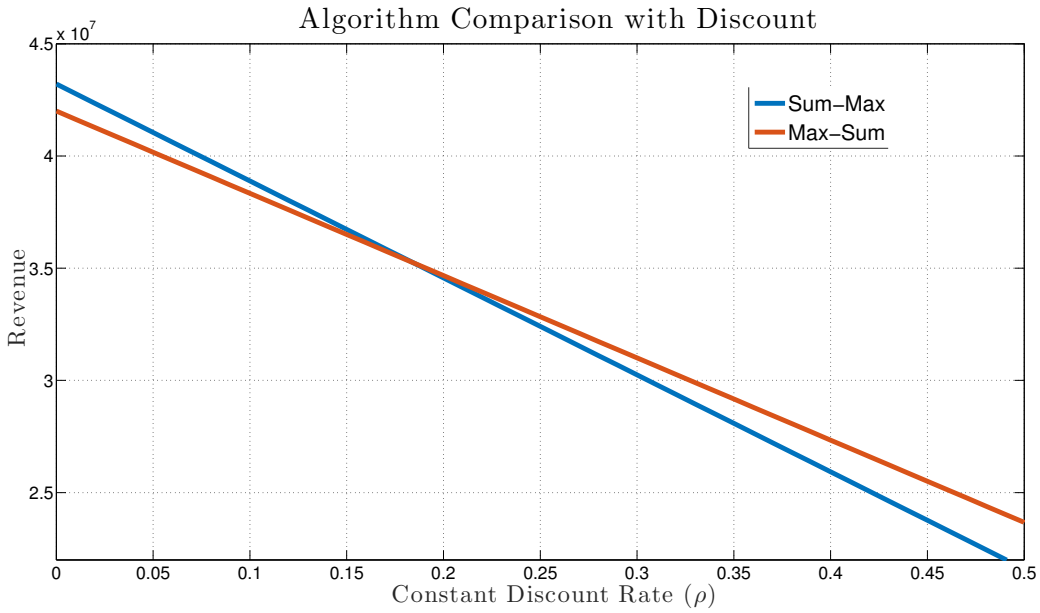
In addition to comparing the revenue, we can consider how many orders were filled and of the orders assigned, how much of that order was actually filled. Because the *Max-Sum* could care less if an order is filled, we find that these orders are filled with an average of 6.30% of the time slots originally requested. Using the *Sum-Max* method, the orders assigned to bidders are filled with an average of 5.82% of time slots originally requested. In fact, using the *Sum-Max* method the maximum proportion of filled orders is only 53.53%. Also, in the *Max-Sum* method, one order is completely filled, but no orders are completely filled in the other method. Furthermore, using the *Max-Sum* method 75 bidders are left with no time slots assigned to them, but the *Sum-Max* method leaves only 46 bidders with nothing. We conclude that *Sum-Max* is very efficient in generating revenue, but it is not very friendly since it fills no orders completely. The table below summarizes the results.

Value Matrix	Method	f	f_r	Avg % Filled	Zero Assigned
P^*	*Sum-Max*	\$43,491,328.11	0.3062	6.30%	46
	Max-Sum	\$39,212,490.31	0.2755	5.82%	75
Q^*	*Sum-Max*	\$27,496,495.03	0.1936	2.89%	123
	Max-Sum	\$23,762,822.66	0.1673	5.55%	168

Using the *Max-Sum*, we can try to maximize revenue with different criteria for selecting the 'best' bidder in each step of the algorithm. We tried several methods, which included the following: highest bid per time slot value, minimum Hamming distance (described below), and minimum slots remaining from original order. The last method is motivated by the fact that it may be beneficial to fill any orders that are nearly complete. Every one of these methods had a smaller f_r as compared to the 'maximizing revenue' criterion. In fact, $f_r < 0.23$ for these methods.

Effect of Discounts

Although we didn't have quite enough time to implement a discount scenario into the algorithm, we were able to conclude that if a discount were implemented into the *Max-Sum* algorithm, it could potentially yield higher revenue values. This is due to the fact that no orders are completely filled using the *Sum-Max* method. If we use this method, no customer is completely satisfied, which means we cannot take their original bid because there is no way to satisfy it. In this case, we would have to give every buyer a discount. In contrast, the *Max-Sum* method satisfies at least one customer. This means we do not have to give that customer a discount. To demonstrate this scenario, we could assume (a rather extreme assumption) that we give a 100% discount to anyone who does not get their complete order. In this case, the *Sum-Max* method would yield a revenue of zero and the *Max-Sum* method would still yield a non-zero value. To demonstrate this further, we consider a discount ρ if your order is not completely filled. Then we can see that if a discount larger than $\rho = 19\%$ is applied, then the *Max-Sum* method actually yields a higher revenue.



This indicates that if there is a discount involved, then it may be beneficial to test both methods. This further demonstrates that keeping some costumers happy may have a positive influence on the overall revenue.

3 The Effect of Pods

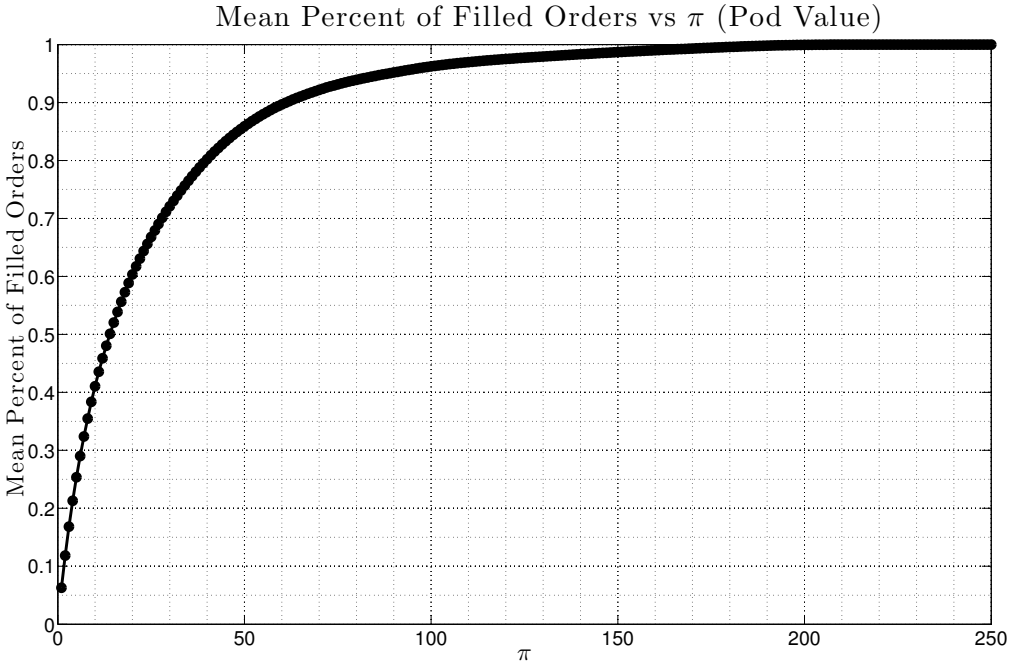
Some time slots are longer than others. Because of this, a time slot may contain several pods or smaller portions of one time slot that we can fill with several buyers' commercials. In this sense, we say that π is the number of pods that can be filled in any given time slot. For simplicity, we assume that π is constant for each time slot. An acceptable value for π is probably around 4 or 6 because a time slot of length 120 seconds may be filled with four 30-second commercials or six 20-second commercials. To demonstrate the effect of a varying pod value π on the performance of our *Sum-Max* algorithm, we consider the the proportion of time slots that are completely filled with commercials. To compute this last metric, we define the Hamming distance for any bidder as

$$d_i = \sum_{j=1}^{n} |b_{ij} - c_{ij}|.$$

Since $C = [c_{ij}]$ is the assignment matrix, the value d_i is the number of time slots that are not filled after we assign bidder i a time schedule. Furthermore, we can compute the proportion of time slots originally requested that are filled after assignment as d_{ir} given by

$$d_{ir} = 1 - d_i \left(\sum_{j=1}^{n} b_{ij} \right)^{-1}.$$

To get an idea of how this value changes with π, we plot the average d_{ir} over all bidders using the results of the *Sum-Max* model. Consider the graph below:



This graph makes perfect sense. Since π is the number of pods available in each time slot, we conclude that we can more easily fill more time schedules as π increases. In fact, as π approaches the maximum number of time slots in a bidder's schedule, we'd expect d_{ir} to approach 1. In our data, the maximum number of time slots in a bidder's schedule is 216. Therefore, if $\pi = 216$, then we would theoretically be able to fill everyone's order. This is why the graph asymptotically approaches 1 as $\pi \to \infty$.

Algorithm Engineering

Erik Palmer
University of South Carolina

Marilyn Vasquez
George Mason University

August 9, 2016

The purpose of the Algorithm Engineering Subgroup was to design and analyze the approach our group took towards determining how bids should be awarded. This involved taking a high-level view of the overall process, identifying how to split tasks so they could be more efficiently coded, determining the elements necessary for each module to better fit into the algorithm, and asking questions about the implications of our approach. This subgroup did not write any code nor run any tests. Instead, we outlined steps which could later be coded into part of the overall algorithm.

Identifying the overall goal as an iterative combinatoric auction, we divided the algorithm into three main parts: constructing the P-Value Matrix, the winner determination problem (WDP) and a counteroffers and discounts phase. Within this framework we discussed two approaches for constructing the P-Value matrix. The winner determination problem was covered by other subgroups and was therefore not a focus for us. In the counteroffer and discount portion of the algorithm, we identified multiple ways that counteroffers could be constructed, and outlined ideas for implementing seller side discounts.

1 The P-Value Matrix

The driving force behind most of the algorithm is the assumption that we know what each slot is worth to a bidder, even when they do not bid on it. This information is what is coded into the P-Value matrix (denoted as P elsewhere). In this subgroup report we will focus on how the P-Value matrix was viewed within the problem and mention how it could be changed or expanded to incorporate additional features.

In order to approach the problem a simplifying assumption was made for the P-Value matrix. While both demographic and advance target data is available, the matrix was constructed from only demographic considerations using the weighted sum method. From our point of view the advanced targets could be thought of as added resolution for the

demographic data. In this sense, the 30 categories of demographic data would be increased by the number of categories of advanced targets. Since all parts of the overall algorithm should work for any number of categories, we decided to omit advanced targets in our considerations.

In addition, it should be noted that a true bid originates as a constraint based order. However, we considered a buyer's bid price with chosen time slots as the starting point for this algorithm. Therefore it is possible that with additional information from the constraint based order, the P-Value matrix could be further improved to more accurately reflect the buyer's unstated prices for each time slot. It should also be pointed out that since the determination of the P-Value matrix depends only on each buyer's bid, it could be computed in parallel.

1.1 Other Methods for Creating the P-Value Matrix

Identifying the P-Value matrix as an independent module, allows for substituting other possible calculation methods without modifying the overall algorithm. An example of such a drop-in replacement, is the mean comparison method for identifying the most important demographic features to a buyer. As this method will be explained in detail in another subgroup's write up, we will not discuss it further here. Another possibility would be to combine both demographic and temporal methods, to adjust the calculation of initial P-Values to more heavily favor the demographics of the slots in which earlier impressions would appear. The details of this adjustment to the P-Values are described in the Temporal Weighting subsection.

1.2 Questions Regarding the P-Value Matrix

There are a few questions that we considered about determining the P-Value matrix and its place in the algorithm. The first question is whether the P-Value matrix should be modified with each iteration of the auction? This has some important consequences as modules developed later in the algorithm, such as the counteroffers and discounts, could be considered to alter the value of each slot in the eyes of the bidder. In this sense, we should ask whether the initial P-Value matrix should be weighted more heavily then subsequent iterations, or whether iterations on the P-Value matrix should converge to the actual price the buyer is willing to pay for a certain combination of time slots.

1.3 Temporal Weighting

It was pointed out that the three first impressions are the most important to bidders. Therefore a scheme which accounts for this temporal difference was devised to represent this in the values of the P-Value matrix. The scheme proceeds as follows:

1. Assume slots are arranged in time, from first to last showing. (This assumption is not necessary, only an ordering.)

2. Calculate the P-Value matrix as usual, based on demographic data.

3. Re-weight the P-Values so that the first three impressions are given more weight and hence higher value.

To illustrate this idea, suppose the weights are 1/2, 1/4, 1/8 for the first impression second impression and third impression respectively, and the final 1/8 is divided between all remaining slots equally. We could categorize this as a 2^{-x} example. Then with these weights we can use the same method of rescaling to determine a new bid that adjusts the perceived value to the buyer upwards in the first slots. The most appropriate temporal weighting scheme would need to be derived from historical data or guided by seller experience.

The following is an example of temporal weighting using the example bid in the table:

Bids	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
Bid 1:\$1150	1	0	1	1	0	0	1	1
Bid 1 P-Values	\$300	\$0	\$400	\$200	\$0	\$0	\$100	\$150

Assume the slots are ordered in time. We associate the largest weight, 1/2, to Slot 1 because it is the first impression, 1/4 to Slot 3, the second impression, and 1/8 to Slot 4, the third. The remaining 1/8 is divided evenly between all the leftover slots. Thus slots 7 and 8 both have weight 1/16.

Next we define a few terms in order to simplify the calculations and elucidate the example. They are B, the bid price, $\boldsymbol{P}$, the price for each time slot, and $\boldsymbol{w}$, the weights. The symbol $\circ$ denotes the piecewise product of two vectors. Let

$$\begin{aligned} B &= 1150 \\ \boldsymbol{P} &= \{300, 0, 400, 200, 0, 0, 100, 150\} \\ \boldsymbol{w} &= \{1/2, 0, 1/4, 1/8, 0, 0, 1/16, 1/16\}. \end{aligned}$$

Then to determine the scaling factor α we use

$$\alpha = \frac{B}{\boldsymbol{P}\boldsymbol{w}^T}. \tag{1}$$

The following equation determines the P-Values updated with the temporal weighting scheme:

$$\boldsymbol{P'} = \alpha \left(\boldsymbol{P} \circ \boldsymbol{w} \right). \tag{2}$$

Using these methods, this example results in the following revised P-Values:

Bids	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
Bid 1 P'-Values	\$593.55	\$0	\$395.70	\$98.92	\$0	\$0	\$24.73	\$37.10

Here we can see that the temporal weighting shifted the bidder's perceived value towards their first three impressions, thereby achieving the desired result.

2 The Winner Determination Problem

The winner determination module of the algorithm is the most computationally intensive. If we simply award each time slot on the basis of who we expect will pay for the most for it (after exceeding minimum reserve), then the algorithm is relatively straightforward. Unfortunately, this typically results in no one getting their entire order and thus a lot of theoretically unhappy customers. The aim of the algorithmic engineering subgroup was to isolate operations so they could be separated into different modules of the algorithm. In this sense, we view the approach to the winner determination problem as either the *Sum-Max* or *Max-Sum* algorithms. Since these approaches are discussed in detail elsewhere we will go no further here.

2.1 Giving Preference to Certain Buyers

As was discussed in other subgroups, there exist reasons to give certain buyers preferential treatment. The goal for the algorithm engineering subgroup was to identify these reasons, and think about how they could be incorporated into the algorithm. Moreover, we wanted to foresee conflicts and ask meaningful questions that arise from taking particular approaches.

The main question that arose from considering preference for certain buyers was whether we could put a dollar amount on these preferences. This arises from the fact that taking the *Sum-Max* approach yields the highest revenue for the seller. Therefore any preference for certain buyers would necessitate reduced revenue. So the question becomes how much revenue are we willing to sacrifice in a single auction, and in what situations, in order to give buyers preferential treatment?

In consultation with the WDP subgroup, three methods for identifying good customers that could be given preferential treatment were outlined. They are: 1. Those bidders closest to having their entire order filled. Mathematically, this is the minimum Hamming distance between the bid and accepted offers, and is discussed further in other subgroup summaries. 2. Those bidders who offer the most above the minimum reserve measured on a slot by slot basis. 3. Other customer satisfaction reasons, such as owing the bidder a favor, or desiring to establish or keep a relationship with a certain bidder.

One theorized method for incorporating this preferential treatment was to add the determined monetary value of the preferential treatment to the P-Value Matrix. This would raise the theoretical value a buyer was offering for a time slot. For example, a bidder close to having their entire order filled might have the P-Values increased $20 across the board. This change would make the algorithm more likely to award them the time slots. However when the auction completed, the actual revenue taken in by the seller would be only the original bid, and not the theoretical value of the preferential increase.

2.2 Tie Breaking

A comprehensive method for breaking ties is necessary. Both *Sum-Max* and *Max-Sum* methods are likely to encounter situations where bidders are offering the same amount. Ties could be broken using a number of measures: 1. In the case of *Sum-Max*, we could consider which bidder has the largest total order. 2. In the case of *Max-Sum*, we could consider which bidder has the least number of desirable slots. Desirable slots would be a slot in which another buyer would be willing to pay more for that specific slot. 3. On whether the buyer is bidding on a subset of their original order. 4. Other preferential measures for buyers.

2.3 Seller Price Vector

In order to isolate modules, simplify the winner determination problem and allow for greater flexibility in building different types of counteroffer and discount schemes, the idea of a seller price vector arose. This vector would represent the price at which the seller will sell each time slot. At the initial stage of the auction, it could start as the minimum reserve for the seller for each time slot. It would then be updated during each iteration of the auction. That is, if a buyer offered more for a particular slot than the minimum bid or another bidder, the seller price vector would be increased the respective P-Value amount for that time slot. Having a vector like this would allow the algorithm to track a minimum price which must be surpassed in order for the seller to agree to sell the time slot. Please note that since this seller price vector mechanism was considered late, it was never coded into the project.

3 Counteroffers and Discounts

The last major section of the algorithm determines the sellers response to the buyer's bids. As was clear from simulations of the WDP, it is almost certain buyers will not be able purchase their first choice at the first price, and some type of ensuing negotiation will be necessary. Therefore we aimed to incorporate mechanisms into the algorithm which would guide this next step in the auction process. Due to the multiplicity of options that could be considered when responding to a failed bid, this portion of the algorithm has many more independent modules than other parts. By design, these responses could be combined or discarded without impacting the overall algorithm structure.

Each of the following subsections explains how a counteroffer or discount type mechanism might work. We started by dividing them into the categories: replacement, subset, minimum increase and discounts. Within each of these categories small variations of implementation could result in large differences in the outcome of the auction. In general, replacement refers to offering time slots other than what was included in the buyer's bid. Subset refers to offering only a part of the original bid. Minimum increase to win outlines how to determine how much a buyer should increase their bid to get what they want. Finally, the discount here refers exclusively to seller side discounts: that is, the seller deciding to discount a time slot in order to make it more attractive to buyers.

In addition to outlining how these modules could be coded, we also tried to look ahead to consequences. In particular, the effect of offering the minimum bid increase to win seemed to need the most theoretical framework to avoid leading to unrealistic results. However, as with all the techniques explained, thoughts on how to adjust, or mitigate altogether, any undesired behaviors in the algorithm are given.

In considering the sections below, we often thought of responding to results from the simplest WDP, taking the maximum of each time slot column (*Sum-Max*). While not stated, some of the sections will be far more challenging to understand as part of an algorithm using another method for the WDP. Another simplifying assumption for all modules in this phase of the algorithm is that buyers do not compete directly with each other. Rather they compete only with the winning bidder. Competition between bids is incorporated as the algorithm iterates and a new winning bid is chosen.

3.1 Demographic Similarity

The first type of counteroffer we considered was termed a replacement type. That is, for time slots with losing bids, other time slots we believe the buyer would be interested in are offered with the prices at which the seller would accept. The question then became how to determine which time slots to offer as replacements. As we assumed the demographics of impressions was the largest factor influencing the buyer's decisions, it was natural to compare the demographics of remaining time slots. The details of how to determine demographic similarity between two time slots is the subject of another subgroup.

A simple outline of a potential demographic similarity replacement algorithm is written here:

1. Identify slots with losing bids.
2. Calculate combined demographic distribution of losing bid slots.
3. Search for combinations of unassigned slots possessing a similar distribution, order results in terms of similarity.
4. From unassigned slots, identify slots which have a seller price closest to the original bid.
5. Rank these offerings based on the demographic similarity and price closest to original bid.
6. Return counteroffer.

In this outline there is still a large amount of flexibility in implementation. For example, in step 2, rather than focus on the demographic distribution of losing bids, we could go by what demographics were decided to be most desirable to the buyer at the beginning of the algorithm. Another example would be to limit the replacement slots offered in step 4 to a

combination that remained under the seller's original bid. In this sense, counteroffers would always be within the constraints of the buyer's original budget. Conversely, a modification to step 4 could allow a combination of available slots which have similar demographics, but possibly larger audience, to be identified and offered at prices which exceed the buyer's original bid. This type of counteroffer could be considered as a replacement upgrade.

3.2 Subset Offer

The subset offer is the simplest of all the counteroffer schemes we considered. In the subset offer, any slots requested by, but not won by the bidder are removed from their bid. The price of the counteroffer is then computed by subtracting the P-Values corresponding to their bids on all the losing time slots. This type of counteroffer only results in less slots awarded to the buyer and less revenue to seller.

The scheme is outlined below:

1. Identify time slots with losing bids.
2. Remove them from the buyer's order, lowering the buyer's bid by the P-Value for each lost slot.
3. Return counteroffer containing only the winning bids at a reduced price.

3.3 Minimum Bid Increase to Win

An important type of counteroffer is for a buyer to know how much they should increase their bid to get exactly the time slots they want. This is beneficial to the seller because it raises the amount the seller will receive for a specific time slot. Two ways to incorporate the minimum bid increase to win counteroffer were considered. They are increasing the bid based on a time slot by slot basis, or increasing based on the P-Value weighting of the entire bid. The latter method led to unreasonable increases in the bid for a buyer. Nonetheless, it led to some interesting mathematical questions which were examined by other subgroups. The first method, which examined each slot individually, led to more realistic values.

A direct slot by slot method of determining the minimum increase looks at a bid in which the buyer did not get all the requested slots. Then it would identify the difference between the winning bid's price for that time slot and the losing buyer's price. If there was no other bid on the slot, and the buyer's bid was below the minimum reserve, we could think of the minimum reserve as the winning bid amount. The next step would be to calculate the minimum amount needed to win by summing these differences for each time slot. The counteroffer would then consist of the buyer's original bid increased by the minimum bid to win.

The minimum bid increase to win algorithm outline is as follows:

1. Identify slots with losing bids.

2. Sum the difference in P-Values and the seller's price.
3. Increase buyer's bid amount by this difference.
4. Return counteroffer.

In considering the implications of using the algorithm described above, we discovered a scenario in which this method would be less than optimal. Consider the case when a buyer is overbidding on one time slot but whose P-Value for the second slot is below the seller's price:

	Slot 1	Slot 2
Bid 1: \$1500	\$1000	\$500
Seller's Price	\$500	\$750

Since the seller's price is the maximum amount the seller will receive for the time slot, it's clear that accepting the bid as is, would lead to an overall price higher than the combination of seller prices for the two time slots. However, the minimum bid increase to win algorithm would currently ask the buyer to pay an additional \$250 to win the bid. This seems unreasonable since the buyer is currently overpaying on Slot 1. To avoid this it seems a modification to the algorithm is necessary.

One possible modification would be to divide the bid into two different portions. The part that matches the seller's price, and the amount that pays above that. Then the amount that pays above the seller's price could be redistributed as necessary to other time slots. In the current example, the buyer is overpaying for time slot 1 by \$500. Suppose we split that between the two slots, \$200 for the first, and \$300 for the second. Then the bid would be revised as follows:

	Slot 1	Slot 2
Bid 1: \$1500	\$700	\$800
Seller's Price	\$500	\$750

Here we can see that since each P-Value is above the seller's price, the minimum increase would be \$0, and the seller could accept the bid outright. Furthermore, if this was not the case, and the bid for a desired slot was still below the seller price after redistributing the overpayment we would return to original minimum increase to win algorithm.

This modification to the minimum increase algorithm is outlined as follows:

1. Identify losing slots.
2. Identify slots with overpayment.
3. Redistribute overpayment to slots with losing bids.
4. Do the Minimum Increase algorithm.

3.4 Iterative Seller Discount

A mechanism to include discounts offered by the seller in order to sell less desirable slots was considered. We theorized that within the iterative auction framework, the seller's price for each time slot could be reduced a fixed amount at each step. The discount could also increase as some sort of exponential function. Either method would serve the same goal of generating revenue for the seller from slots that would otherwise not be awarded.

3.5 Final Counteroffer or Discount

The question was asked about what would happen at the end of the algorithm to time slots for which the minimum bid was not met. In considering this, we thought about the final iteration of the time constrained algorithm. It seemed reasonable that at the final counteroffer phase, bids for time slots should be awarded even if the P-Values were below the seller's minimum reserve. While it would not be helpful at any other point in time, doing this on the final step would result in revenue from time slots that would otherwise be zero.

4 Conclusion

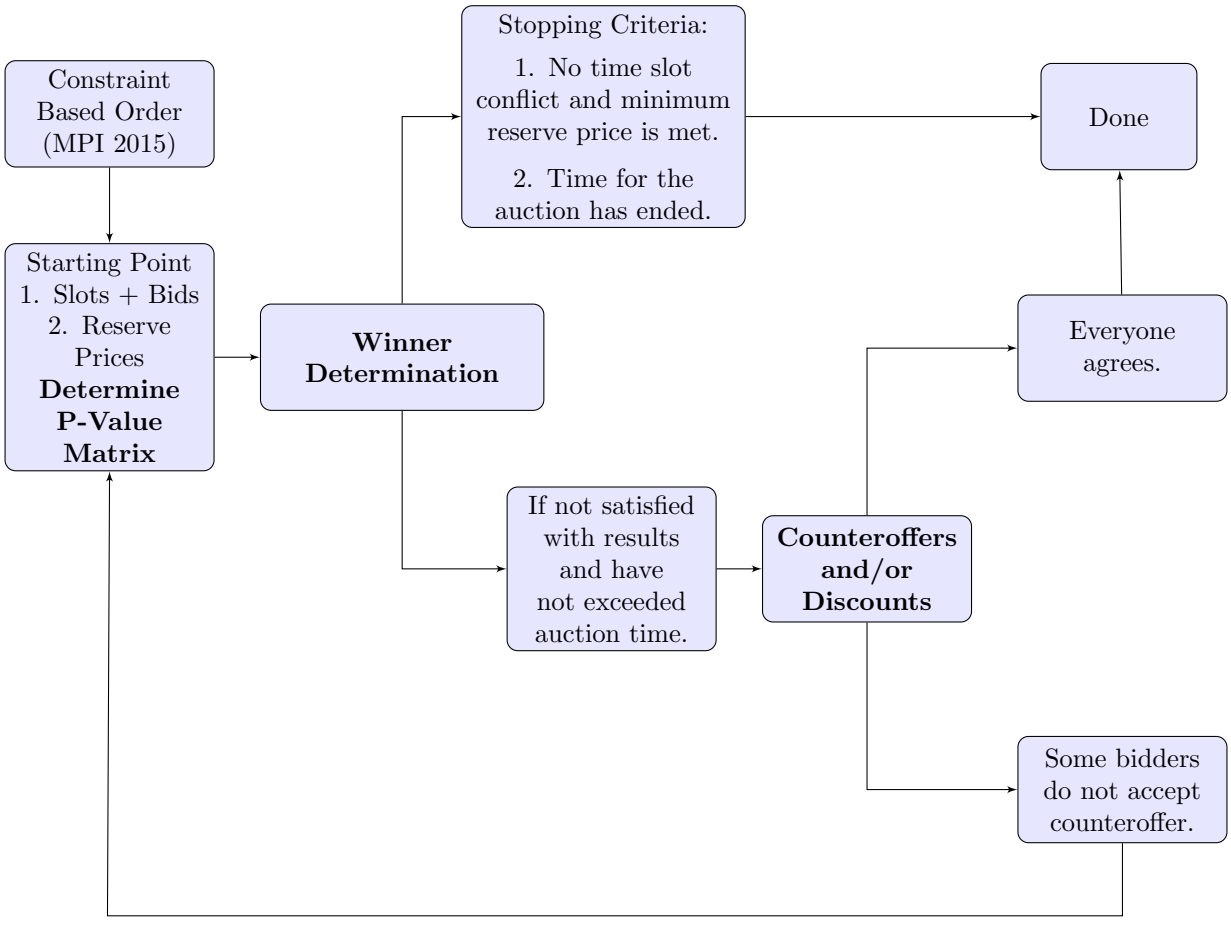
The algorithm engineering subgroup's holistic approach to the problem tied together many of the different subgroups. As such, many of the ideas presented above arose out of discussions with other subgroups. Our main contribution was to build a framework to combine all the parts which were examined separately. This involved identifying and examining scenarios in which our algorithm led to unreasonable outcomes, as well as proposing general guidelines for addressing these issues and expanding the capabilities of the algorithm.

We concluded that the optimal approach was to organize the algorithm into three phases: determining the P-Value matrix, the winner determination problem, and the counteroffers and discounts portion; occurring in that order. This translates to determining what each buyer will pay for each time slot, who each time slot is awarded to, and then how to respond with counteroffers and discounts for unawarded time slots. Within each of these self-contained phases a variety of smaller modifications could be considered. For example, we outlined a temporal method for altering the P-Value matrix. In the second phase, we identified methods to incorporate additional considerations into the winner determination problem, such as preference for certain buyers, tie breaking, and tracking the seller's price. Finally, in the counteroffers and discounts phase, we discuss a multitude of methods with outlined processes to calculate reasonable counteroffers. In taking this structured approach we formed a robust framework for which many further modifications to the iterative algorithm could be conveniently included.

Algorithm Flow Chart

Illustration by Melike Sirlanci

July 13, 2016



A Mathematical Programming Approach to Solve the Programmatic TV Advertising Problem

Pavan Kumar Narayanan
SUNY Buffalo State
pavan.narayanan@gmail.com

August 9, 2016

This report summarizes the models and methodology that have been developed as part of a solution architecture for the combinatorial auction problem for hybrid programming TV markets. This solution is attempting to solve merely a part of master problem and should be combined and looked at along with Dr. Edwards' master solution methodology. I intend to keep my methodology succinct and have divided this section into two parts: description about part of the problem that we have attempted to solve using mathematical programming and the proposed methodology that was executed with data provided by Clypd. Please refer to the problem statement given by Clypd for a detailed presentation of the problem statement.

1 Problem Structuring

The problem is to determine the best strategy for TV programmers to balance constraint-based and bid-based orders in order to maximize profit and the number of accepted orders. In auctioning advertising slots, at each round of the auction, the bidders would select more than one slot (group of items where each item corresponds to a slot) which they wish to purchase and place an order for the entire package that consists of multiple placements. The TV programmer's objective is to maximize the profit and number of accepted orders. One of the constraints is that, at the end of each round of the auction when the placement is allotted to a specific bidder, it is deemed a completed transaction. Once a TV placement is allotted to a specific bidder, that placement will be removed from the available list of TV slots for business.

In the context of mathematical auction theory, this problem can be formulated as a combinatorial sealed-bid auction where the bidder submits bids for combinations of items in each round while the seller (TV Programmer) would look for efficient allocation of bids in order to maximize the revenue. The latter allocation problem is called the Winner Determination Problem.

To illustrate, let us look at the problem scenario:

Table 1: Placements Schedule

Placements ID	TV Network	Date/Time
A	CBS	2016-06-15 20:00:00
B	Fox	2016-06-15 20:00:00
C	PBS	2016-06-15 20:00:00
D	ABC	2016-06-15 20:30:00

Let us look at the potential bids for each combination of slots:

Table 2: Potential bids

S	Bidder 1	Bidder 2	Bidder3
AB	50	10	10
BC	10	40	**60**
CD	10	10	50
AD	60	**70**	20
ABC	60	30	60
BCD	40	50	30
ADC	40	60	50
ADB	30	50	40
ABCD	100	100	100

We can see that allocating placements A and D to Bidder 2, and allocating B and C to Bidder 3, would maximize the TV programmer's revenue and satisfy all the orders. The complexity arises when we have hundreds of bidders, each bidder bidding more than one combination of items for thousands of placement slots. The data provided by Clypd had 300 bids for 14861 placements.

2 Solution Methodology

The following is the high-level solution methodology employed to solve this problem. By employing this methodology, we could generate revenue of $174,697,013.30 and fulfill 33.91% of orders in the first round of the auction.

Step I: Solve the Master Problem and Sub-problem:

(a) *Master Problem:*

$$maximize \sum_{j=1}^{n} P_j X_j$$

subject to

$$\sum_{j=1}^{n} B_{ij} X_j \ \leq \ T, \ \forall \ i \in m,$$

$$X_j \in \{0, 1\},$$

where

$$P_j \ \leftarrow \ \textit{the price vector that corresponds to bids,}$$

$$B_{ij} \ \leftarrow \ \textit{the 'i' number of items in each bid j,}$$

$$T \ \leftarrow \ \textit{number of items that could be fit in the slot.}$$

The objective function is to maximize the revenue by choosing the highest-priced bids subject to the constraint that, while choosing the bids, the total number of placements for each time slot should be less than or equal to the total number of available placements, T.

(b) *Sub-problem for assigning # of placements to each time slot T:*

$$\textit{maximize} \ \sum_{i \in A} \sum_{j \in T}^{n} C_{ij} Y_{ij}$$

subject to

$$\sum_{i \in A} Y_{ij} = 1, \ \forall j \ \in \ T, \ \textit{(time slots),}$$

$$\sum_{j \in T} Y_{ij} = 1, \ \forall i \ \in A, \ \textit{(chosen placements),}$$

$$Y_{ij} \in \{0, 1\}.$$

Here, each time slot T can accommodate several placements; for example, a time slot T of 120 seconds could handle four ad-placements of 30 seconds each. This assignment problem further looks for opportunities to enhance total revenue generated by further letting the bidders choose placements that would help them reach out to their target audience effectively.

Step II: Upon obtaining the optimal solution to the above problem, remove the accepted bids from the bids matrix to form a new bids matrix:

$$B_{ij} \ - B^*_{kj} \ = B_{i-k,j} \ \leftarrow \ \textit{new bids matrix.}$$

Step III: If a bidder bids for a placement that is less than the reserve price, then remove that bid (now we reshuffle and propose new bids to bidders who lost in the first round of the auction):

$$\textit{if } P_j \ < \ r_j \ \textit{then } B_{ij} \ = \ 0, \ \forall \ i \in \{1, 2, 3, ..., 300\} \ (\textit{bidders}).$$

Step IV: For each time slot, choose the maximum bid price that a bidder has proposed:

$$\arg\max_{j} (j_1, j_2, j_3, ...j_n), \ \forall \ j \ \in \{1, 2, 3, ..., 14861\} \ (\textit{time slots}).$$

Step V: Extract the maximum bid prices and transform to obtain the TV programmer's proposed allocation for each bidder. For illustration,

Bidder 14 alloted slot # (336, 1148, 1201, 9935, 10000)
Bidder 16 alloted slot # (325, 1207, 9856, 9926, 11477, 11530, 11755, 11785)
Bidder 28 alloted slot # (2316, 7558)
.
.
Bidder n (....)

Step VI: Repeat Steps I to IV for the next rounds of the auction.

The idea is to propose the revised allocation for bidders that would help the bidders themselves to choose other bids that are not allocated yet in the second round of the auction.

3 Scope and Limitations

3.1 Scope of this Methodology:

- **Recommendation System Development:** From the TV programmer's point of view, while proposing the revised bid allocation to bidders, a recommender engine may be developed by looking at similarity measures between viewers of various demographics that would benefit the TV programmer's unallocated slots and help bidders explore new time slots.

- **Time Slot Assignment Problem:** The assignment problem can be modeled with increased complexity as the time slots may not be of the same duration. Some slots can be of 120 seconds duration, and there could be cases where smaller or larger durations may arise. TV placement assignment should benefit the TV programmer by further maximizing revenue by satisfying more bidders in the initial round of the auction.

3.2 Limitations of this Methodology:

- **Order Target:** Order targets for bidders who did not win the first round were only partially met. Ideally, the problem statement is to meet order targets for all bidders but due to constraints, we can only satisfy high-priced bids. This may not help the bidder-TV programmer relationship if bids were not met consistently due to low-priced bids.

- **Replacements:** Replacements are when a bidder fails to win an time slot, a different item that is available and closest to the target demographics may be offered. Replacements are not included in this methodology.

4 Conclusion

Although there are numerous ways to solve these sorts of problems, the mathematical programming approach may be the best way of attempting to solve this combinatorial auction problem. This solution methodology may be looked at as a first step to solve the problem and there is certainly enough scope to satisfy all the bidder's requirements by recommending certain time slots that would fit a given bidder's target demographics, and offering replacements with competitive pricing that would satisfy the bidder's requirements. We should also collect relevant data about pricing, the nature of the business, and many more attributes about the bidder in order to further develop this proposed methodology.

Qualitative Properties of the Value Matrix

Marina Chugunova

July 12, 2016

Assume that due to a winning strategy the ability of a buyer to get a particular time slot in a bid depends on a corresponding estimated value in the value matrix P. That means that if the computed estimated values p_{ij} of the buyer i for the time slot j is not big enough the buyer might not be able to get the desired time slot (someone else can take the slot j). It seems quite obvious that the buyer should not be able to reverse the situation (*i.e.*, to get the slot j) by just adding more time slots to the bid without changing the total payment.

Definition 1.1 *We say that the value matrix P is **consistent** if the estimated value p_{ij} of each time slot j in the bid of the buyer i will not increase if an additional time slot is added to this bid while the total bid price stays unchanged.*

Let demographics, which is used to construct the value matrix P, consist of $k = 1, \dots, K$ different categories. For example: if we have data about three age groups for males and females then the total number of different categories is given by $K = 3 * 2 = 6$. Let N be the total number of time slots available in the bidding game, *i.e.*, $Slot_j = 1, \dots, N$. Consistency of the value matrix P does not depend on a particular ordering of the time slots so, without loss of generality, we can compare for the bidder i two options: to bid for the slots $Slot_j = 1, \dots, N-1$ or to bid for all slots $Slot_j = 1, \dots, N$ using the same total bid price. If the value matrix P is consistent then bidding for an extra slot N should not increase estimated values of time slots $Slot_j = 1, \dots, N-1$.

Using that the demographics distribution for each time slot $Slot_j$ is known and given by positive integers $Slot_{j,k}$, we can represent the non-increasing property of the estimated values of time slots $Slot_j = 1, \dots, N-1$ by using the following inequality (where all $Slot_{j,k}$ with $j = 1, \dots, N$, $k = 1, \dots, K$ are positive integers):

$$\frac{\sum_{k=1}^{K} Slot_{j,k} \left(\sum_{n=1}^{N-1} Slot_{n,k} \right)}{\sum_{k=1}^{K} \left(\sum_{n=1}^{N-1} Slot_{n,k} \right)^2} \geq \frac{\sum_{k=1}^{K} Slot_{j,k} \left(\sum_{n=1}^{N} Slot_{n,k} \right)}{\sum_{k=1}^{K} \left(\sum_{n=1}^{N} Slot_{n,k} \right)^2}.$$

The inequality above is derived by straightforward computations of the estimated values of the $Slot_j$ in the value matrix P for two types of bidding: for slots $Slot_j = 1, \dots, N-1$

and for slots $Slot_j = 1, \ldots, N$ with the same total bid price (because the total price is the same in both cases it actually scales out from the inequality, *i.e.*, the consistency is solely defined by the given demographics data).

To get some insight into the consistency property let us start from a set of simple examples (particular cases).

1. Example 1 (the demographic data available is only showing the total number of people watching TV at the given time interval $K = 1$).

   The inequality simplifies to a trivial one:

   $$\frac{Slot_{1,1}}{\sum\limits_{n=1}^{N-1} Slot_{n,1}} \geq \frac{Slot_{1,1}}{\sum\limits_{n=1}^{N} Slot_{n,1}},$$

   that holds true for any positive integers.

2. Example 2 (only two time slots are available in the bidding game $N = 2$).

   The inequality simplifies to a trivial one:

   $$1 \geq \frac{\sum\limits_{k=1}^{K} Slot_{1,k} \left(\sum\limits_{n=1}^{2} Slot_{n,k} \right)}{\sum\limits_{k=1}^{K} \left(\sum\limits_{n=1}^{2} Slot_{n,k} \right)^2},$$

   that holds true for any positive integers.

3. Example 3 (only three time slots are available in the bidding game $N = 3$). To simplify notations in this example we will use $j = 1$, $a_k := Slot_{1,k}$, $b_k := Slot_{2,k}$ and $c_k = Slot_{3,k}$ so in new variables the inequality can be rewritten as:

   $$\frac{\sum\limits_{k=1}^{K} a_k(a_k + b_k)}{\sum\limits_{k=1}^{K} (a_k + b_k)^2} \geq \frac{\sum\limits_{k=1}^{K} a_k(a_k + b_k + c_k)}{\sum\limits_{k=1}^{K} (a_k + b_k + c_k)^2},$$

   where a_k, b_k and c_k are positive integers. Unfortunately this inequality fails if c_1 is small and b_2 is large compared to all other coefficients. The simplest way to see why the inequality does not hold in general for $N = 3$ is based on application of basic calculus. If this inequality holds then it holds for all nonnegative real numbers. Indeed, due to homogeneity (because it holds for positive integers) it would also hold for all positive rational numbers and then we can proceed by continuity. Therefore if we denote by $F(\mathbf{a}, \mathbf{b}, \mathbf{c})$ the right hand side of the inequality, then if the inequality holds then $\partial_{c_k} F(\mathbf{a}, \mathbf{b}, \mathbf{c}) \leq 0$ for $\mathbf{c} = 0$. Calculating the partial derivative $\partial_{c_1} F(\mathbf{a}, \mathbf{b}, \mathbf{0})$ for $K = 2$ we can see that it becomes positive as $b_2 \to \infty$ that leads to the contradiction. The inequality fails for example for the set $c_1 = 1, b_2 = 40, c_2 = b_2 = a_1 = a_2 = 10$.

A more interesting question, from the applied point of view, is the optimal choice of the time slots for the constrained maximal payment. Assume that a buyer cannot pay more for the bid than some fixed maximum amount of money (budget constraint). If the buyer adds to the bid too many time slots then estimated values for these time slots might become too low and this will result in losing the desired slots in the bidding game. Reducing the number of time slots in the bid will increase the probability of getting them. The open question here could be the question of the optimal strategy for the buyer under this money constraint (to avoid the situation "if you are too greedy you might get nothing").

Replacements

Irene De Teresa
University of Delaware

Marilyn Vazquez
George Mason University

July 25, 2016

As previously described, there is a very large percentage of unsatisfied buyers, so it is in the main interest for the sellers to develop a method to provide counteroffers and replacements that are attractive for the unsatisfied buyers but are still profitable. The aim of this section is to propose methods to find replacements that offer a similar number of population impressions in the categories that were originally interesting for the buyer.

1 Clustering Whole Orders

One of the methods to find replacements consists on first mapping the set of possible orders $\mathcal{U}$ into a new space (called the *Demographic space*), where the demographic properties associated to these orders are better reflected. Having done so, we proceed by clustering the orders using these *demographic coordinates*, to finally identify the orders that are in the same cluster that the unsatisfied buyer originally wanted. Among the orders that are in the same cluster (and presumably with more similar demographic characteristics), we choose the ones that are still available and, if possible, that have some of the originally requested slots. The following steps describe this idea more precisely.

Step 1. Let N be the number of available slots offered by the seller. We first find a space of orders

$$\mathcal{U} := \{b \in \mathbb{Z}_2^N \mid b \text{ is a possible order of slots}\}.$$

Here

$$b_i = \begin{cases} 1 \text{ if slot } i \text{ is included in the order,} \\ 0 \text{ otherwise.} \end{cases}$$

Naturally, since in realistic examples the number of slots available in a week is typically around $N = 14000$ and the number of slots selected by the buyer is around $k = 100$, one

can't be exhaustive and include all the $\binom{N}{k}$ possibilities. Therefore, the first problem that we face is how to construct $\mathcal{U}$ in a way that includes enough "possible orders".

In order to solve this issue, we propose to take the original set $\mathcal{B}$ of orders included in our database, which consists of 300 real orders from the buyers, and then run the following algorithm:

$\mathcal{U} = \mathcal{B}$

for all b in $\mathcal{B}$:

$b_y := \{i \,|\, b_i = 1\}$

$b_n := \{i \,|\, b_i = 0\}$, $l_n := |b_n|$

for all i in b_y

$\delta = b$

$\delta_i = 0$

$p \sim \text{UNIF}(1, ..., l_n)$ % p is a random variable in $\{1, 2, ..., l_n\}$

$n^* = n(p)$

$\delta(n^*) = 1$

$\mathcal{U} = \mathcal{U} \cup \{\delta\}$.

end

end

Step 2. Once $\mathcal{U}$ is defined, we can use the demographics matrix V to map this set of orders into a space that reflects the total number of impressions per population category that the buyers intend to reach.

More precisely, we define the map $f : \mathcal{U} \subset \{0, 1\}^N \to \mathbb{N}^n$, by

$$f(b) = b * V^T,$$

where n is the number of population categories (which typically is $n = 30$), and $V \in \mathbb{N}^{N \times n}$ is the demographics matrix: that is, $V_{i,j}$ is the number of impressions of population category j that are associated to slot i.

Step 3. This third step consists of identifying the different clusters in the range of f, in order to associate possible orders that share similar distributions of impressions per population category.

Step 4. (The replacement) Finally, in order to find a replacement for a concrete b that couldn't be satisfied, we identify the cluster C_b to which $f(b)$ belongs. Then, cancel all the possibilities in C_b that conflict with any already-assigned slot, defining:

$$C'_b := \{f(b') \in C_b \,|\, b' \text{ does not conflict with any already-assigned slot}\}.$$

And finally, we choose as a replacement of b the element $b_r \in \mathcal{U}$ defined by

$$b' \text{ such that } f(b') = \arg\min\{||f(b) - f(b')||_2 \,|\, b' \in C'_b|\},$$

where $||\cdot||_2$ is the usual Euclidean norm in $\mathbb{R}^n$.

Toy example. In order to give a concrete example, we will consider now a very simple case. Let's consider $N = 6$ slots and $n = 2$ population categories (male and female), with associated demographics matrix to be the transpose of the following table:

V^T	M	F
Slot 1	30	10
Slot 2	25	05
Slot 3	25	20
Slot 4	05	35
Slot 5	10	25
Slot 6	05	30

Let's suppose that we are in the situation where one of the buyers bid for an order u_L, but that this cannot be satisfied since another already-assigned order u_w has slots in common with u_L. So we can start now our replacement algorithm.

Step 1. Let's suppose that the buyers typically choose two slots among the six available. In this toy example, we can consider $\mathcal{U}$, the space of possible orders, to be exhaustive: that is, $\mathcal{U}$ consists of all the $\binom{6}{2} = 15$ vectors of length 6 with either 0 or 1 in each entry:

$\mathcal{U}$	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6
b_1	1	1	0	0	0	0
b_2	1	0	1	0	0	0
$\vdots$						
b_{15}	0	0	0	0	1	1

Step 2. Now we transform all the possible orders in $\mathcal{U}$ into the demographics space. In Figure 1, we show the image of these 15 orders under the mapping f.

In Figure 2 we show the two conficting orders u_L and u_w. Therefore, we proceed in the algorithm.

Step 3. At this point, we first find the cluster to which the unsatisfied buyer belongs in the demographics space (Range(f)) (Figure 3). Next, we discard orders in this cluster that possibly conflict with the assigned slots (Figure 4), to finally find the option that is closest to b' (Figure 5).

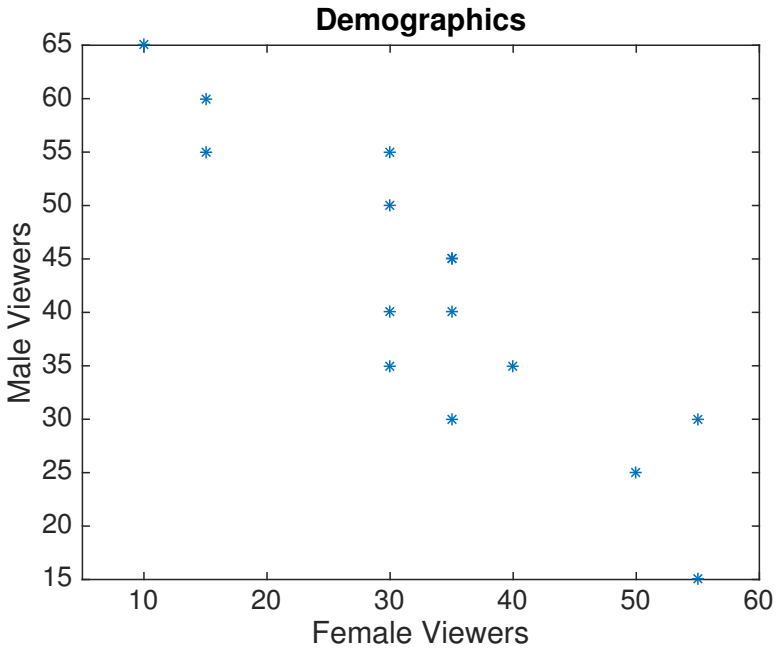


Figure 1: Mapping $\mathcal{U}$ into the *demographic space* (step 2 of the algorithm). Each of the points $f(u) = (f_u, m_u)$ marked here corresponds to the demographic values associated to order u (that is, f_u and m_u are the total number of female and male viewers associated to the slots in u).

2 Clustering Individual Slots

An alternative to clustering complete orders is to cluster individual slots. If a buyer wants slots that were already assigned to someone else, the seller can offer slots in the same cluster as the ones the buyer originally requested. The following is an outline of the implemented algorithm to achieve this goal:

Input: $\mathcal{B}$, V, W (indices of winners)
Step 1. Identify L (indices of non winners), F (indices of a slot no longer available), and R (indices of a slot that needs a replacement).
Step 2. Cluster the rows of V: *i.e.*, cluster the slots based on their demographics.
Step 3. For $i \in R$ and $k = 1, \ldots, N$:

$$C_{i,k} = \begin{cases} i & k \text{ is the index of a slot in the same cluster as } i, \text{ and } k \notin F, \\ 0 & \text{otherwise.} \end{cases}$$

Step 4. Form $S \in \mathbb{N}^{m,N+1}$, where $S_{1,j}$ tells you the bidder index and for $\ell \in L$ s.t $\ell \geq 2$, $S_{\ell,k} = C_{\ell,k}$.
Output: S

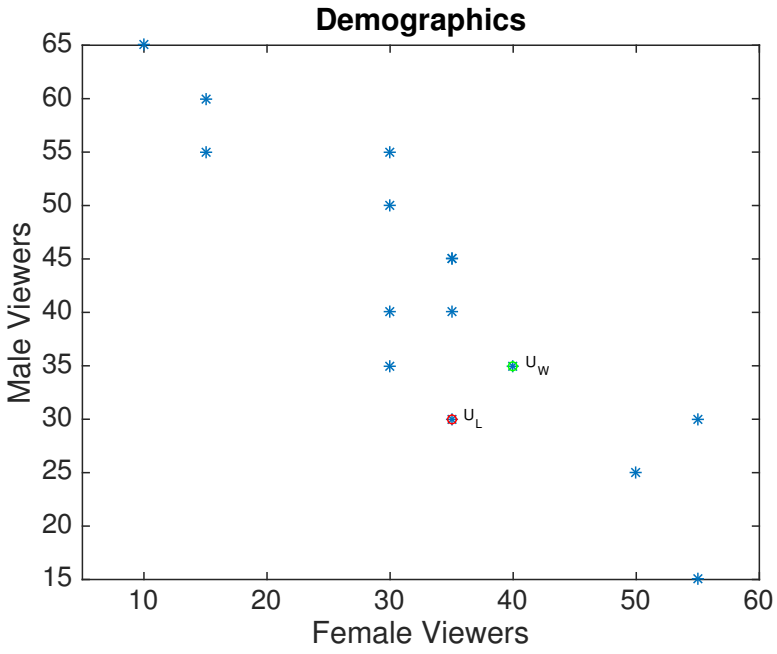


Figure 2: In the *demographic space*, the two conficting orders corresponding to the unsatisfied buyer and to the winner are shown: u_L (shown in red) and u_w (in green).

To illustrate the algorithm, consider an auction with 5 bidders ($m = 5$) and 75 possible slots ($N = 75$) with the orders in Table 1.

Bidder	Slots requested
A	5,10,12,20,25,52,55
B	5,10,20,32,45,49,62
C	5,10,17,20,31,61,72
D	14,51,60,63,65,74,75
E	11,44,54,60,64,65,75

Table 1: Sample orders of 5 different bidders.

For simplicity, suppose that the demographics of interest are only gender viewership and that the first 50 slots are mainly viewed by females and the last 25 slots are mainly viewed by males. Figure 6 shows how this distinction forms clusters in the data. Of course, with real data, we expect noise and high dimensionality to represent some of the challenges to the clustering step.

Now, suppose that after the bidders give the seller their orders and corresponding bids, the seller decides that Bidders B and D are the winners, so $W = \{2, 4\}$. For the first step in the algorithm, it is easy to calculate that $L = \{1, 3, 5\}$. Also, from Table 1, we

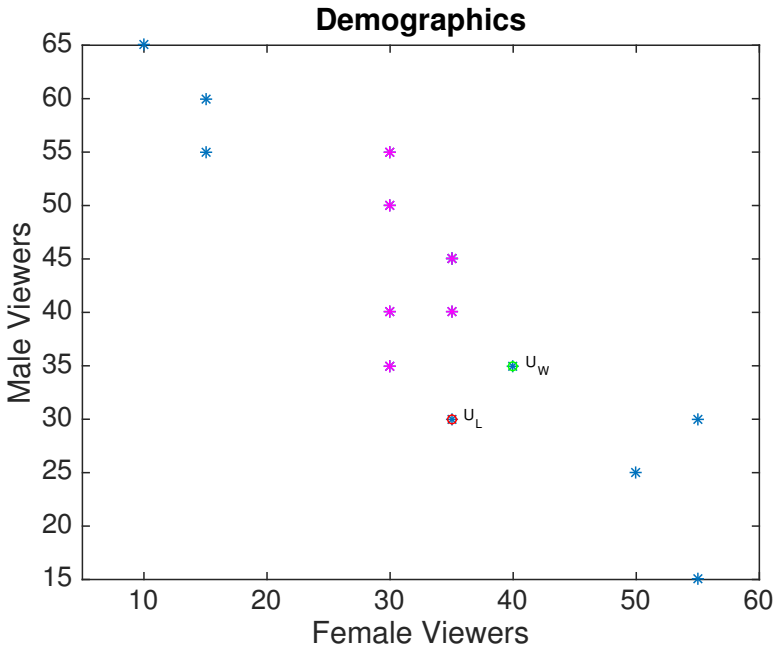


Figure 3: Step 3. Identification of the set of orders u that belong to the same cluster as u_l (shown in pink).

get that $F = \{5, 10, 11, 12, 14, 20, 25, 32, 44, 45, 49, 51, 52, 54, 55, 60, 63, 64, 65, 74, 75\}$, which is the array containing the slots that have been given away. Looking at Table 1 one also can see that slots 5, 10, and 20 were given to Bidder B, and slots 60, 65, and 75 were given to Bidder D, but the other bidders also wanted some combination of these slots, which means that $R = \{5, 10, 20, 60, 65, 75\}$. Step 2 is illustrated in Figure 7(a), where the colored points in represent the slots taken, and specifically the yellow dots represent the slots that need to be replaced. The clustering step, which is step 2 of the algorithm, can be seen in Figure 7(b). One can think of step 3 as finding the replacements and keeping a record of which slot they are replacing. For our example, Figure 7(c)-(d) shows the suggested replacements made by the algorithm, which is done in step 3. In Figure 7(c) one can see how the suggestions, which are marked in blue and green, are the available slots in the same cluster as those that need to be replaced. Figure 7(d) is another illustration of the results, where the x-axis represents the slots that need replacement and the y-axis represents the slots that will be given as replacements. For example, we see that (10,1) is a green circle and (70,55) is a blue circle. This means that slot 1 can be a replacement for slot 10, which was in the second cluster and hence the green color. Similarly, slot 55 can be a replacement for slot 70, which was in the first cluster and hence the blue color. The red stars in Figure 7(d) represent those slots that were not available as replacements. Note that these correspond to F. On step 4, we simply add in the first column of S the information of which bidder needs the replacements that we gave. In our example, that just means that we identify Bidders A and C as the ones

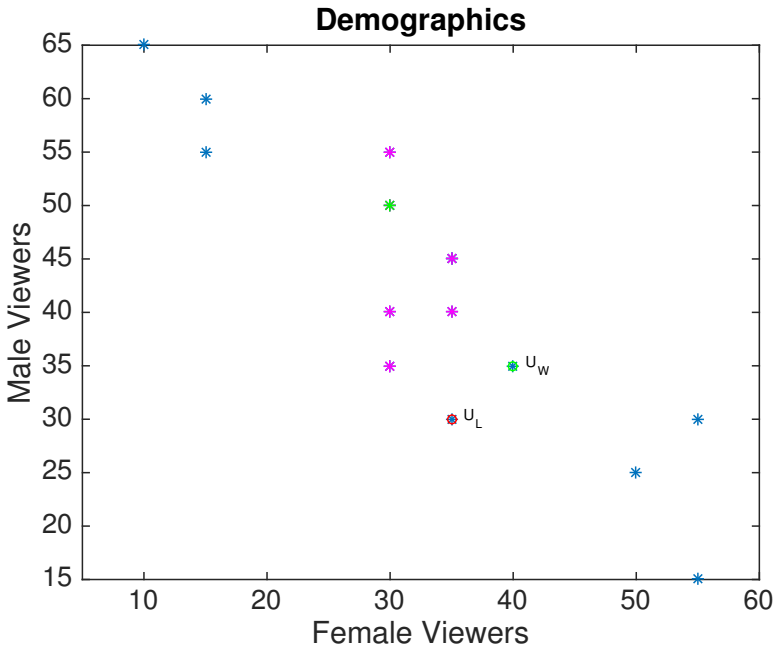


Figure 4: Step 4. In green, we mark all the orders that we cannot take into account for replacements because they contain already assigned slots.

that need the replacements for slots 5, 10, and 25 and Bidder E as the one that need the replacements for slots 60, 65, and 70.

3 Future Work and Challenges

One of the main issues of both these approaches is which clustering algorithm to use. While the user can chose the clustering algorithm of their choice, we are using the continuous k-nearest neighbors algorithm in [1], since in previous research by some of the authors, this algorithm has shown to be efficient and accurate.

Future work includes: (1) test these algorithms with real data and calculate the accuracy and efficiency of both approaches and (2) narrow down the suggestions for each bidder. (1) is important because it will help us better understand the benefits and costs of each algorithm and help the user decide the algorithm that best suits their needs. To accomplish (2), we can minimize the Euclidean distance in the demographic coordinates, but we must also consider the bidder's budgets, willingness to pay for each slot, and the reserve price.

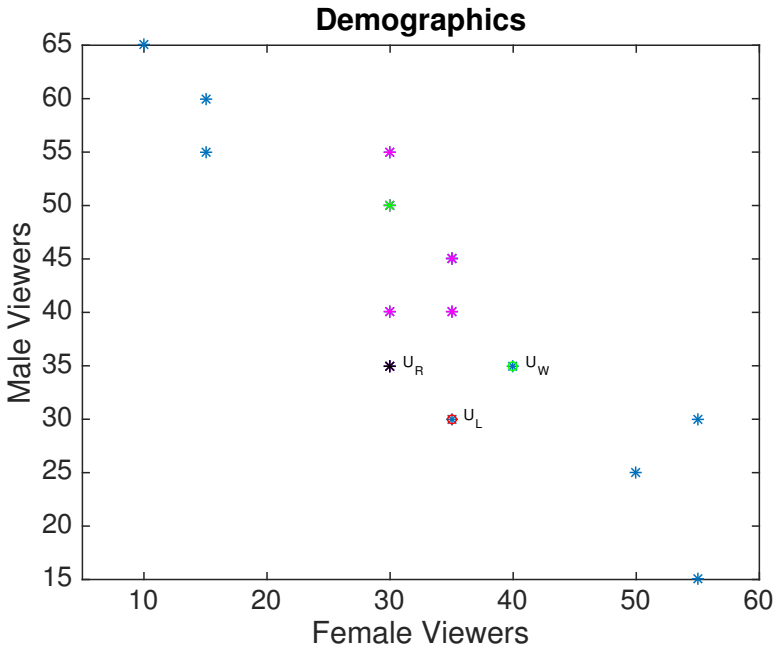


Figure 5: Among the options in the same cluster that don't conflict with any already assigned slot (still shown in pink), the proposed replacement is the closest to u_l.

References

[1] T. Berry and T. Sauer. Consistent manifold representation for topological data analysis. Submitted to Foundations of Computational Mathematics, 2016.

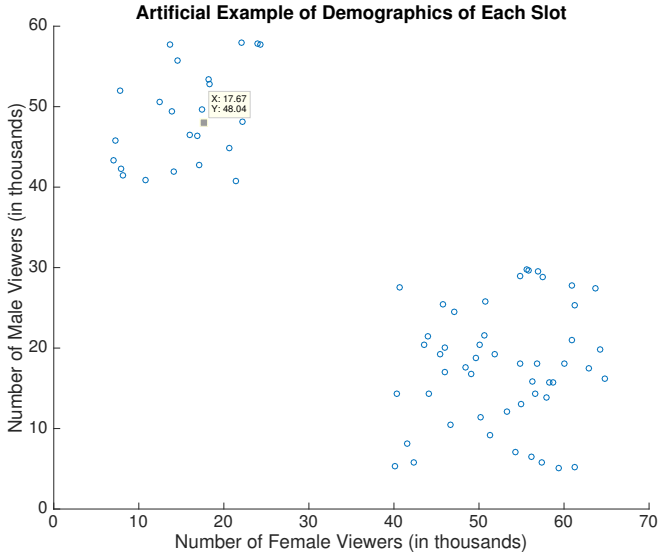


Figure 6: Each data point in this figure represents the demographic coordinates of a single slot. For example, slot 57 has 17.67 thousand female viewers and 48.04 thousand male viewers.

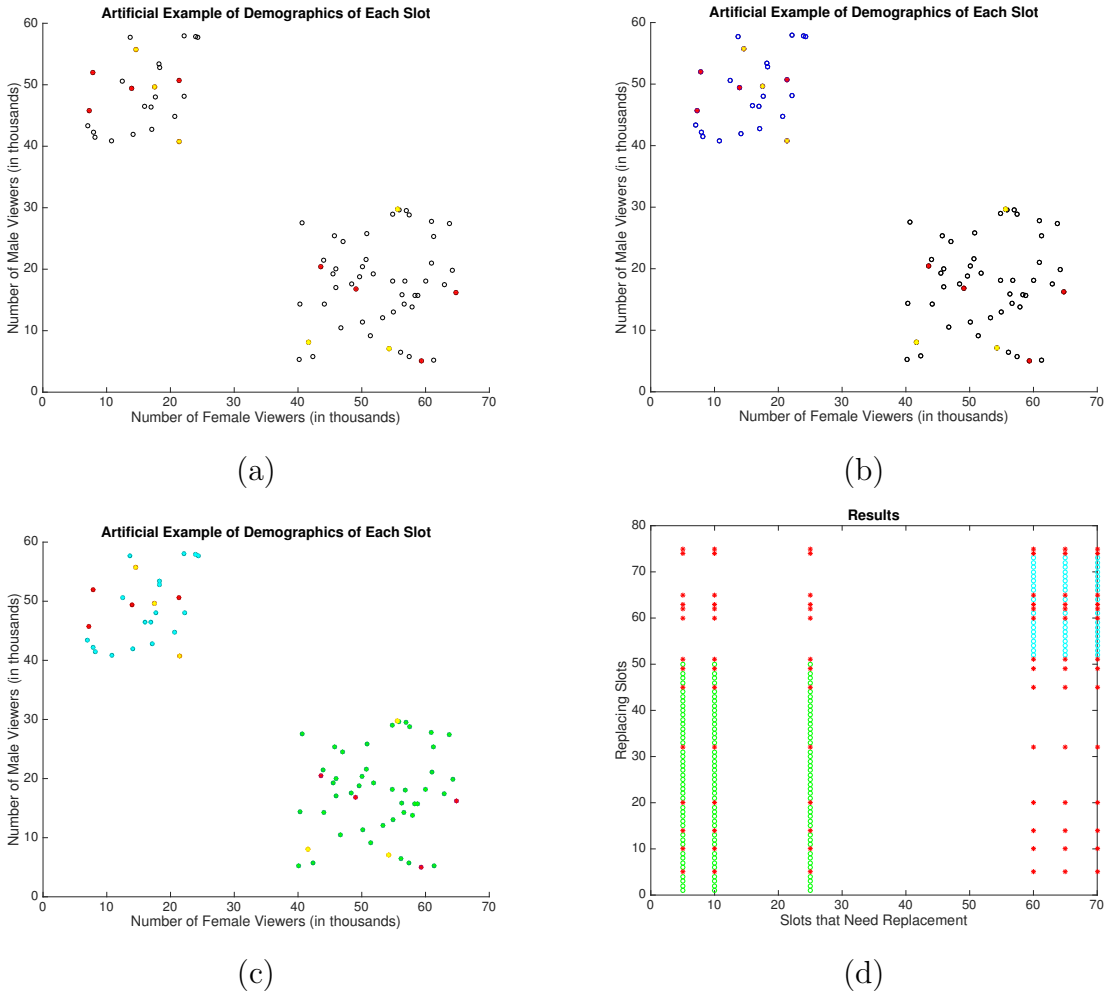


Figure 7: (a) Step 1: The colored data points, both yellow and red, represent the slots that were taken by the winner of the bid. In particular, the yellow data points are those slots that were requested by bidders that did not win the auction and hence the seller needs to find replacements for them. (b) Step 2: The data clustered. (c) The suggested replacements in each cluster. (d) Another way to view the results. The x-axis represent the slots that need replacement (5,10,20,60,65,70) and the y-axis the slots that can replace them. The red stars are those slots that were already taken.