Modeling and Optimization of Reach and Exposure in TV (clypd Problem)
MPI 2017 Report

Tishara Collins, Delaware State University
David Drzewicki, Rensselaer Polytechnic Institute
Peter R. Kramer, Rensselaer Polytechnic Institute
Qingxia Li, Fisk Unversity
Weifan Liu, Duke University
Yun Lu, Kutztown University
Zack Morrow, North Carolina State University
Pushpi Paranamana, Texas Tech University
Christopher Raymond, University of Delaware
Abiy Tasissa, Rensselaer Polytechnic Institute
Mehdi Vahab, Florida State University
Yuri Yatsenko, Houston Baptist University

June 16, 2018

Problem Presenter: Marco Montes de Oca
Other Industrial Representatives: Margaret Colberg, Jingsong Cui

1 Introduction

The scheduling of advertisements during television shows involves a marketplace with various television networks selling advertising slots at various times to advertisers. Roughly speaking, television networks will announce an inventory of available advertising slots over some future time horizon (say a 1 week period) with known programming schedule, and advertisers will place bids in the form of a desired number of impressions made on a targeted demographic of viewership, together with a budget and other side constraints. The demographic distribution of impression for a future advertising slot can be estimated by Nielsen data collected for previous episodes of the program during which the slot would air, as well as for programs within the same genre or time of day. A company such as clypd can serve as a sort of broker for the television networks by taking in the bids from various advertisers and trying to set an advertising schedule that is responsive to the details of the requests by the advertisers and which makes good use of the available inventory of advertising slots on the networks. Many aspects of the advertising bids are actually linear in nature, and a constraint satisfaction algorithm can be used to try to find a feasible solution which satisfies all advertising requests, or to report a scheduling solution which violates some wishes of the advertisers as minimally as possible. The computational complexity of such a linear constraint satisfaction algorithm will be considered to be low enough as to be practically implementable.

One aspect of viewership that is nonlinear and challenges such a constraint satisfaction algorithm is the *exposure distribution* of an advertising campaign scheduled over multiple slots. The number of *impressions* I on a given demographic group made by an advertising campaign is simply the (statistically estimated) sum of the number of people from that demographic group viewing each slot within the campaign. The exposure distribution e_j, by contrast, refines this simple impression count by characterizing the distribution of how many people within the target demographic group viewed the advertisement exactly j times within the campaign, for $j = 1, 2, \ldots$ The total number of impressions is a simple summary statistic related to this exposure distribution by $I = \sum_{j=1}^{\infty} j e_j$. Another fundamental summary statistic of the exposure distribution is the *reach*, the number of unique individuals who viewed an advertisement within the campaign at least once: $R = \sum_{j=1}^{\infty} e_j$. The reach of an advertising campaign is much more difficult to compute than the number of impressions, because it is sensitive to the relationship between the viewership of different advertising slots. By contrast, the number of impressions is a marginal statistic that can be evaluated by considering each slot separately of other slots, and summing.

The retroactive estimation of reach of an advertising campaign can be done rather straightforwardly from detailed Nielsen viewership data by taking the union of the viewership of each time period during which an advertisement was aired. For a campaign involving $N^{(a)}$ advertising slots, though, this computation of the union of $N^{(a)}$ large sets (of Nielsen viewers) is considerably more expensive than the computation of a sum of $N^{(a)}$ integers. Moreover, because the reach of a campaign involves interactions between the viewership of different advertising time slots, it is fundamentally nonlinear in nature. These two observations create significant challenges for scheduling advertising campaigns which take into account reach constraints or goals. Suppose for example that we take advertiser bids involving purely linear constraints, and now seek to optimize the reach of the various campaigns while respecting the constraints of the bids and inventory. Such an objective could be used by clypd to the enhance the value of television advertising to the advertisers, keeping television advertising attractive and therefore the client commericial networks happy and operational, and thereby retarding to some degree the development of advertising modalities featured in “Minority Report.” The nonlinearity of reach as a function of schedule requires a substantial algorithmic development to the linear constraint satisfaction approach described in the opening paragraph. A naive approach which attempts to calculate reach exhaustively for all scheduling choices of a single advertising campaign of $N^{(a)}$ ads over 10^4 possible slots would require on the order of $10^{4N^{(a)}}$ reach calculations, each of which is relatively expensive as noted above. Efficient, theoretically grounded, general purpose approaches to the constrained optimization of nonlinear functions in high-dimensional state spaces do not appear to be readily available. Mathematically founded nonlinear optimization algorithms almost always exploit some special structure of the objective function, such as convexity. But as we shall see in Section 2, reach as a function of schedule does not appear to exhibit convexity or any other helpful properties (other than non-negativity). Nonlinear optimization of objective functions with complex structure, such as reach, is often approached through heuristics.

The working group at the Mathematical Problems in Industry (MPI) workshop centered on the question of how the optimization of reach to desired demographic groups could be incorporated in a practically computable fashion into a scheduling algorithm. To keep focus on this challenge, we excluded from consideration several other relevant issues. First of all, we only consider a single advertising bid, with specified linear constraints, and do not consider at all how to reconcile conflicts between the scheduling of campaigns from different advertisers over the same time frame. (The schedule conflict resolution issue was examined in the 2016 MPI workshop). Secondly, we will assume the availability of adequate data or a statistical forecast model to characterize all information of interest regarding the demographic viewership of the slots available for ad placement, together with overlap of viewership between slots. As a concrete test case to exclude the statistical forecasting issue, we can envision trying to retrospectively optimize the reach of an advertising campaign over a previous time period for which Nielsen viewership data is available.

Because of the complex structure of reach as a function of schedule, we confined our attention to standard classes of optimization algorithms that are relatively simple to implement and inexpensive to execute. Indeed we took computational efficiency as a central concern, and sought more to improve signficantly the reach of an advertising campaign relative to the output of a constraint satisfaction algorithm, rather than to truly optimize reach (an apparently unattainable goal in practice due to the computational complexity). In particular, no algorithm we propose involves the true calculation of reach at all; they rather refer to reach through more easily computable surrogates. We begin in Section 2 by stating the scheduling optimization problem in more precise mathematical terms, and raising a question concerning how the slack variables are represented in the modified objective function for a linear constraint satisfaction algorithm, which neglects reach issues, that was presented in a technical report [de Oca, 2017]. We then proceed to present three approaches to including reach considerations into a scheduling algorithm.

One idea (Section 3) is to remain in a linear programming framework through the addition of a linear objective to the linear constraints. On the face of it, a linear function cannot represent reach, which fundamentally refers to interactions between the viewership of scheduled slots, but these interaction effects are taken into account through the choice of coefficients of the linear objective function. Clustering analysis of slots, based on viewership overlap between slots, is used to deflate the coefficients corresponding to slots that contribute a relatively small number of impressions within a large cluster. The deflation of coefficients and solution of the linear programming problem can proceed iteratively, to reduce the value of slots with high viewership overlap with already scheduled slots, while the iterated solutions are improving reach. A second approach (Section 4) is to modify the constraint satisfaction problem through the introduction of a quadratic objective function, the maximization of which, under linear constraints, should still amount to a computable linear problem in high dimensions. This quadratic function, which is supposed to be a simple representation of reach, has positive linear terms representing the number of impressions made with each scheduled ad, and negative quadratic terms that subtract out estimates for repeat viewership of ads scheduled in two different slots. Finally, we considered greedy algorithms (Section 5) for solving optimal set coverage problems, which builds a schedule slot-by-slot without "thinking ahead." Essential to a greedy algorithm is the formulation of a function "to be greedy about" at each iteration. The true reach function was considered still too expensive to use in a greedy algorithm, so we formulated a simplified reach function based on estimates of pairwise viewership overlap between already scheduled slots and potential slots, whose increment when divided by slot cost, would serve as the utility function the algorithm is "greedy about" at each iteration step. The existence of multiple linear constraints to be satisfied by the completed schedule pose a challenge to carrying over the standard set coverage algorithm that is mindful only of a single budget constraint, and we formulated three approaches to managing these side constraints within the greedy algorithm.

2 Problem Formulation

From de Oca [2017], we summarize the scheduling problem as follows. We are given, from the networks, a schedule of $N^{(s)}$ available slots for advertisements, which can be viewed as a grid whose rows indicate the slot index i and whose columns indicate attributes of the slots (hour, day, network channel, program genre, daypart, cost, etc.). Roughly speaking, the slot index is essentially specified by a combination of the hour, day, and network channel, other than noting that the multiple slots available within a given hour will be given separate indices. Other columns give informational attributes about those slots. Advertisers will present a request for ad placements within these various slots, together with budget constraints and other specifications. Now the scheduler must place the campaigns of multiple advertisers over the available slots in such a way as to make the advertisers "happy." Obviously for each slot, only one advertiser can have their ad placed, so there is some conflict in trying to make all advertisers happy if they are targeting similar demographics. This question of conflict resolution was partly addressed in Panaggio et al. [2016], but we leave it out of consideration for the present workshop. Therefore, we simply focus on one advertiser, and the question of how to schedule their advertisements in such a way as to make that advertiser "happy."

The scheduling of a given advertiser's campaign can be described by a binary vector $\mathbf{X} = (X_1, \ldots, X_{N^{(s)}})$, where $X_i = 1$ if an ad will be placed in slot i, and $X_i = 0$ if no ad will be placed in slot i. We remark that of course one could formulate the schedule differently by indexing placement by network and hour (n, h), associating a certain number of advertising slots $M_{(n,h)}$ to each network-hour combination, and defining ad placement variables $X_{(n,h)}$ to indicate how many ads are placed in hour h on network n, which now will be integer-valued variables satisfying $0 \leq X_{(n,h)} \leq M_{(n,h)}$. For simplicity, we will in this report stay with the simpler binary X_i formulation given above, but most of the ideas presented could be carried over with modification to this other version.

With this notation, we can specify two aspects of the advertiser's happiness. One is meeting some explicitly stated demands concerning budget, timing, and types of networks and programs in which to place the ad. As explained in detail in Panaggio et al. [2016] many of these explicit requests can be represented as linear equality and inequality constraints on the schedule vector $\mathbf{X}$. Here we will simply assume these are accordingly specified, and we only have one remark to make about the handling of these linear constraints in Subsection 2.1. Of course some constraints could be formulated in a nonlinear way (such as asking for a certain time interval between successive placements of ads), but such nonlinear constraints will be considered out of the scope of the present workshop.

A second component of the advertiser's happiness is meeting their goals for exposure to their target demographics. To address these concerns, we require a model for the viewership demographic for each slot. This challenging question was partially addressed in Panaggio et al. [2016], but we leave it out of consideration in the present workshop. We shall take the viewership model as given, and focus purely on schedule optimization subject to a viable viewership model. As a practical means of developing the optimization algorithms within this framework, we could imagine scheduling advertising campaigns retrospectively over a time period for which we have Nielsen viewership data, and simply use that data for the viewership model. That would remove inefficiencies caused by inadequacies in the viewership forecasting, and allow the scheduling optimization algorithms to be tested and calibrated apart from these complications. Of course, eventually the scheduling optimization algorithms do have to be coupled to an actual viewership forecast model. For the present workshop, though, we will take as given a model of viewership including overlap between viewership of different slots, for the demographic groups of interest.

For example, suppose we are trying to optimize a schedule over a retrospective time period for which we have Nielsen viewership data. For each of the $N^{(v)}$ Nielsen viewers, we have a weight d_v of viewer v on the target demographic of interest, and for each slot i we have a set $\mathcal{V}_i$ of Nielsen viewers who are watching that slot. This then would give the number of impressions of slot i on the target demographic as:

$$S_i = \sum_{v \in \mathcal{V}_i} d_v. \tag{1}$$

This would be the simplest measure of exposure, and one could formulate an optimal advertiser happiness model by optimizing the total number of impressions on the target demographic $I(\mathbf{X}) = \sum_{i=1}^{N^{(s)}} S_i X_i$ subject the linear equality and inequality constraints. As the objective function $I(\mathbf{X})$ is linear, this amounts to a linear programming problem that we consider feasible to solve.

As discussed in the introduction, another measure of exposure is reach – the number of unique viewers exposed to an advertising campaign. In terms of the scheduling variable and viewership data, it takes a rather complicated form:

$$R(\mathbf{X}) = \sum_{v=1}^{N^{(v)}} d_v \mathbb{1} \left\{ \sum_{i=1}^{N^{(s)}} \mathbb{1}\left\{ v \in \mathcal{V}_i, X_i > 0 \right\} > 0 \right\}. \tag{2}$$

where $\mathbb{1}_A$ is an indicator function giving the value 1 if the condition A is satisfied, and the value 0 if the condition A is not satisfied. The objective function (2) would be rather expensive to compute, and a nonconvex, nonlinear optimization method would have to be employed if we wished to maximize it, subject to the linear constraints. We take this problem as computationally intractable. The purpose of the following sections is to propose computationally tractable, modified optimization problems that try to improve on simply optimizing impressions by taking reach considerations into account, but not literally maximizing the actual reach (2).

2.1 Remark on Linear Programming with Slack Variables

Before proceeding, we pause to report an observation made during the workshop regarding the linear programming examples given in de Oca [2017]. To allow for scheduling solutions even if the advertiser makes demands incompatible with slot inventory, the linear constraints are represented by slack variables. This reformulates these constraints as linear penalty terms in the objective function. Now, the constraints involve certain units, such as megadollars for a budget, or minutes for timing specifications, while the objective function based on the number of impressions made by a campaign has units of kilopersons, for example. Slack variables represent departures of the actual quantity (budget, time, etc.) from the constraint, and so have the same units as the quantities involved in the linear constraint. When the slack variables are incorporated via penalty terms in the objective function, they must be weighted by some conversion factor that, in a sense, is weighing the marginal value of constraint violation against the marginal value of increased impression yield (so, for example, if a certain number of impressions can be made with an at-budget campaign, how many more kiloperson impressions would have to be made by a campaign that is one megadollar over budget to placate an advertiser). As such, the conversion factors have units as well, such as kilopersons per megadollar, or kilopersons per minute. The working group noticed some conversion factors *with different units* were sometimes set to be some numerical multiple of each other. For example, one might want to say that violating the budget constraint is 10 times worse than violating a time constraint. But this statement is not mathematically coherent because the violations in each case have units, so the magnitudes of the quantities depend on the measuring unit. It would be sensible, for example, to say that violating a budget constraint by one megadollar is 10 times worse than violating a time constraint by one minute. But notice that if we were to change units due to a change in scale of the advertising campaign, and now be discussing budget in terms of gigadollars and time in terms of hours, the same relative penalty weighting would amount to saying that violating a budget constraint by one gigadollar is $10^4/60$ times as bad as violating a time constraint by an hour, so in these new units, the conversion factor in front of the budget constraint would be $10^4/60$ times larger than that in front of the timing constraint. We simply wish to stress that if, for example, the conversion factor for the slack variable for timing is called c, and the conversion factor for the slack variable for budget is asserted to be $10c$, the factor 10 should have units associated to it, namely the timing units over the budget units. This consideration would be important if ever the algorithm were run with variables measured in different units, so the conversion factors would be appropriately adjusted by changing their units to the same as those of the slack variables.

3 Iterated Linear Programming Through Clustering Analysis

As noted earlier, the objective function for reach maximization is non-linear and expensive. A candidate surrogate objective function starts with the number of impressions on the desired demographic

$$I(\mathbf{X}) \equiv \sum_{i=1}^{N^{(s)}} S_i X_i, \tag{3}$$

and modifying the coefficients S_i appropriately. Since such a modification is inherently linear, it rarely recovers the solutions of the original nonlinear optimization problem. However, an approach of this nature promotes solutions to a linear program (LP) that maximize reach. Variant of this approach has been used, for example, for finding optimal solutions for the capital budgeting problem [Lusztig and Schwab, 1968].

The ‘appropriate’ modification considered here is clustering of ad slots. To motivate clustering, it suffices to note the obvious fact that loss of reach happens due to repeat viewers. A natural suggestion is clustering of ad slots based on the assumption that people watch shows that are ‘similar’ to each other. The strategy is to group slots with similar viewers together and distribute the ads among the various categories. Consider two slots that are available for a future ad placement. If both slots are on prime time, it is more likely that there is overlap in viewership as opposed to the case where the first slot is in prime time and the second slot is in night time. If the two slots are on channels from networks that are not ‘similar’ to each other, it is less likely that the people who watch the first ad will also watch the second ad. Another important attribute for classification of slots is the genre of the show. If two slots have the same genre, we expect some viewers who will watch ads on both slots. The above considerations suggest that the timing of the slots, the network and the genre are important attributes for the clustering problem. While far from being an exhaustive list, these attributes serve as starting guide for an intrinsic measure of similarity between slots. Given these attributes, the problem now reduces to the general clustering problem. For simplicity, the clustering analysis is done via the K-means method. To transform the categorical variables, e.g slot 10 is shown in night time in network C and has action genre, to a single classifier, the method of one-hot encoding can be used. A similarity metric would also have to be defined to provide a scalar measure for the difference between two slots based on their categorical descriptions. To illustrate how the clustering result can be used to modify the LP objective function, consider an idealized clustering result output shown in the Figure 1, with each color representing a cluster of slots with similar viewership.

The clustering provides us with three clusters of various sizes and diameters. The diameter of a cluster is defined as the furthest distance between any two items within the cluster. Consider the circled slot which

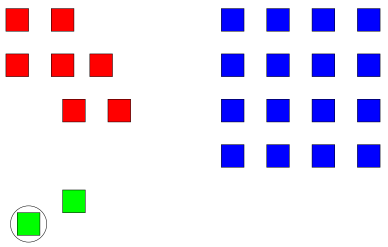

Figure 1: A schematic of clustering of ad slots. Slots in the same cluster are possibly from the same channel, share the same genre and are/or are aired near to each other in time

belongs to the green cluster. Since this slot is in a small cluster, it is less likely to contribute repeat viewers. However, a slot in the blue cluster is more likely to contribute repeat viewers since its viewership overlaps with the viewership of a large number of other slots. Given this, the LP objective should promote slots in small clusters and penalize slots in big clusters. However, care is needed to ensure that the above procedure doesn't penalize slots with large estimated impressions in big clusters, since we may well want to choose the slot within a large cluster that has the greatest number of impressions. To address this, within a given cluster, slots with large estimated impressions will be penalized less than slots with small estimated impressions. To make the above considerations precise, we first introduce some notation. Assume there are $N^{(s)}$ slots. After K-means clustering, slot i is mapped to $C[i]$ where $C[i]$ is the cluster number. The objective is to multiply the coefficients S_i in Eq. (3) by deflating factors β_i with $0 \leq \beta_i \leq 1$ which can depend in principle on cluster size, cluster diameter, distance between clusters, and the relative number of impressions of slots within a cluster. For instance, one possible choice could be:

$$\beta_i = e^{-\gamma_i \dfrac{|C[i]|}{S_i}} \tag{4}$$

where $|C[i]|$ is the size of the cluster to which slot i belongs,

$$\gamma_i = \frac{\|S\|_{C[i],\infty} - S_i}{\|S\|_{C[i],\infty}}$$

measures the relative deficit in impressions of slot i relative to the slot with maximum impressions in its cluster, and

$$\|S\|_{C,\infty} \equiv \max_{i \in C} S_i$$

is the maximum number of impressions made by a single slot in cluster C. The rationale of the choice (4) is to impose a greater penalty to slots that belong to large clusters and have fewer impressions relative to alternative slots within its cluster. This produces an LP with objective function:

$$\tilde{I}(\mathbf{X}) = \sum_{i=1}^{N^{(s)}} \beta_i S_i X_i.$$

After solving this LP and then rounding to nearest integers, we can formulate another LP with modified objective function:

$$\tilde{I}^{(1)}(\mathbf{X}) = \sum_{i=1}^{N^{(s)}} \beta_i^{(1)} S_i X_i. \tag{5}$$

where now $\beta_i^{(1)}$ can further penalize coefficients based on belonging to a cluster $C[i]$ for which the solution of the previous LP gave $X_{i'} \geq 1$ for one or more $i' \in C[i] \setminus \{i\}$ (scheduled an ad to another slot within the same cluster). This LP could again be solved, and the reach of the schedule produced compared to the reach of the previous solution. So long as reach is improving, one could repeat this process of successively deflating coefficients of slots based on their belonging to clusters of other slots scheduled in the previous iteration.

Below, a quick summary of the clustering for LP modification is made.

I. Cluster the slots based on timing, channel and genre.

II. Determine the deflating factors β_i based on the properties of the cluster and number of impressions relative to the cluster

III. Solve the modified LP (5),

IV. From the solution to this LP, rounded to integers, recompute the deflating factors β_i based also on whether the slots are in clusters that have other slots appearing in the previous solution. Iterate while reach is increasing

Another way to do clustering of ad slots is by assigning a weight between any two slots. Consider a graph with nodes representing slots, edges indicating whether two slots are related and a weight informing the extent to which the two slots are related to one another. Given this setup, spectral clustering techniques can be employed. Recent works e.g Szlam and Bresson [2009] have developed a total variation formulation for this problem lending to efficient algorithms. Due to time limitations, spectral clustering techniques were not fully explored and are left for future work.

4 Linear-Quadratic Programming

Since optimization of quadratic functions under linear inequality and equality constraints is still computationally tractable with a large number of variables, we proposed to represent reach via a quadratic approximation:

$$R_1(X) = \sum_{i=1}^{N^{(s)}} S_i X_i - \phi \sum_{i'<i} b_{i,i'} X_i X_{i'} \tag{6}$$

where the first (linear) term is simply the total number of impressions for a proposed schedule, and the second (quadratic) term subtracts an estimate of repeated viewers. This quadratic correction term is more a representation of repeat viewership than an attempt at an accurate estimate, but we may hope its inclusion at least steers the optimization algorithm toward schedules with higher (if not optimal) reach given the prescribed constraints. As with the approximation for the reach function described in Panaggio et al. [2016], the approximate reach function (6) only considers pairwise overlap of viewership between slots. The coefficient ϕ is simply a tuning parameter to adjust the strength of the correction term. One could set $\phi = 1$ to start, and test the optimization on some test problems to see whether a higher value of ϕ (which penalizes viewership collision more) yields schedules with higher reach, or whether ϕ might need to be reduced due to artefacts introduced by too strong a correction term. The coefficient $b_{i,i'}$ is supposed to represent the degree of overlap between viewership of slot i and i', and can be viewed as some function of the properties of the slots i and i', including demographic information about the viewership and time/day of the slot. This function $b_{i,i'}$ could be formulated either via a simple parameterized model or in a purely data-driven manner.

4.1 Parametrized Quadratic Coefficients

Whatever parametric model for $b_{i,i'}$ is selected, it probably should scale linearly with the size of the viewership to avoid having to retune the coefficient ϕ when applied to advertising campaigns with large or niche demographic targets. Indeed, $b_{i,i'}$ is supposed to represent roughly the number of repeat viewers between slots i and i', which one may expect would be proportional to some measure of the viewership of each show.

As an example of a parameterized model of viewership overlap, we could consider

$$b_{i,i'} = \min(S_i, S_{i'}) e^{-\gamma_h |h_i - h_{i'}|} \sigma_{i,i'}$$

where h_i denotes the hour during which slot i occurs, γ_h^{-1} is the mean time a viewer stays tuned to a given channel, and

$$\sigma_{i,i'} = \begin{cases} 1 & \text{if } i \text{ and } i' \text{ share the same network channel} \\ \beta & \text{otherwise} \end{cases}$$

where $0 \leq \beta < 1$ roughly represents the fraction of viewers who watch two given different channels within the same hour ($\beta = 0$ may well be reasonable choice in practice). Note that $0 \leq b_{i,i'} \leq \min(S_i, S_{i'})$, which is desirable in representing, in some rough manner, the number of repeat viewers between slots i and i'. The parameterized model (4.1) has two parameters, γ_h and β that could be chosen based on historic viewership data. This two-parameter model could be elaborated in various ways, such as:

- allowing the mean viewing time γ_h^{-1} to depend on the network channels or even program types associated to slots i and/or i' in question (perhaps the slot with the larger number of impressions),
- replacing the exponentially distributed viewing time model by a different shape that better fits viewership data, even perhaps allowing the viewing time distribution (not just its mean) to depend on the network channel or program type associated to slots i and/or i' (perhaps just the slot with the larger number of impressions)
- allowing the likely overlap indicator $\sigma_{i,i'}$ to depend on channel group, program genre, or number of overlapping program characteristics associated to the slots i and i', including allowing the low-overlap parameter β to depend on the channel or program properties associated to the slots.

The proliferation of parameters in the model should of course be moderated in terms of the data available to estimate those parameters.

4.2 Data-Driven Quadratic Coefficients

A data-driven approach to representing the collision in viewership between slots i and i' would be to choose a certain reasonable number of attributes of slots as relevant, then scan through a historical data set to count the overlap between pairs of slots with those same attribute relationships. As an example, we might consider as relevant attributes: 1) network channel, 2) difference in hour of day, and 3) difference of days. Then, if we have Nielsen viewership data over some time period we think of as appropriate for the time period to be scheduled (perhaps the same time period one year before, or simply a recent time period), we could compute a historical overlap factor

$$\hat{O}(\Delta h, \Delta d, n) = \left\langle \frac{S_{i,i'}^{(2)}}{\min(S_i, S_{i'})} \right\rangle_{\Delta h, \delta d, n}$$

where S_i is the demographically weighted number of viewers in the database who viewed slot i (Eq. (1)), and

$$S_{i,i'}^{(2)} = \sum_{v \in \mathcal{V}_i \cap \mathcal{V}_{i'}} d_v$$

is the demographically weighted number of viewers in the database who viewed both slots i and i', and $\langle \cdot \rangle_{\Delta h, \Delta d, n}$ indicates that the argument is to be averaged over all pairs of slots in the database whose hour in the day differed by Δh, whose day differed by Δd, and which both aired on network channel n. Once this overlap factor is computed from processing the dataset, we can use it to modify the viewership collision coefficient:

$$b_{i,i'} = \min(S_i, S_{i'}) \hat{O}(|h(i) - h(i')|, |d(i) - d(i')|, n(i)) \sigma_{i,i'}$$

where $h(i)$ is the hour of slot i, $d(i)$ is the day of slot i, and $n(i)$ is the network channel associated to slot i. The factor $\sigma_{i,i'}$ is again a factor indicating whether the two slots i and i' belong to network channels or program genres that are likely to have viewership overlap. Naturally this approach can be adapted to other choices of relevant slot attributes. The more slot attributes that are included, though, the fewer pairs of slots in the data set will match particular realizations of those slot attributes, and the noisier the estimated overlap factor $\hat{O}$ will be.

A good way to tune the parameters or functions is to run the schedule optimization algorithm over the same time frame over which the viewership data was available. This would remove the separate question of how well historical data on viewership predicts future viewership, and focuses the model calibration purely on the optimization question (if the future viewership model were perfect).

5 Greedy Algorithms for Viewer Coverage

Intuitively, a greedy algorithm seeks to optimize an objective function iteratively without looking ahead—that is, at each iteration, the algorithm chooses the single slot that adds the most marginal utility. We consider first the problem of maximizing the reach $R(\mathbf{X})$ subject to some maximum cost limit L. We define $\mathcal{S}$ to the set of available slots not yet chosen; it begins as the set of all available slots $\mathcal{S} = \{1, \ldots, N^{(s)}\}$ and is reduced as slots are chosen for scheduling an ad at each iteration step. Based on previous work on the optimal covering problem [Chekuri, 2013], we form a process that, at each iteration, chooses the slot $i \in \mathcal{S}$ that maximizes

$$\frac{M(i)}{c(i)},$$

where $M(i)$ and $c(i)$ are, respectively, the incremental reach and incremental cost associated with choosing advertising slot i.

5.1 Implementation on Historical Nielsen Data

As with some of the other approaches attempted during the workshop, we found it useful to test proposed algorithmic approaches to schedule advertisements retrospectively with reference to actual viewership data. With this viewership data in hand, one could directly implement the above greedy algorithm idea, with each step choosing the ad slot with the most additional reach to the target demographic. As an illustration, we sought to enhance reach for an advertising campaign over one month with a target demographic of women in age group 18 - 35. We imposed advertiser-specified constraints that the ads should be at least 3 hours apart, involve no more than 1000 showings, and fit within a budget of \$150,000.

A greedy algorithm was implemented on the sample database provided for clypd regarding Nielsen viewers of various TV channels, together with advertising inventory and costs over the same period. 99,870 viewers from this database fell within the target demographic, and only views of at least 15 minutes were considered. The slot with the largest number of impressions on the target demographic was chosen as the starting point, with further ads constrained to be within 15 days of this optimal target impression slot. The viewers already reached by this first slot are removed from further consideration. Then the next slot for an ad is chosen which makes the largest number of impressions on the remaining viewers not yet reached within the target demographic, while respecting advertiser-imposed scheduling constraints. This greedy approach can be iterated until the budget is exhausted.

5.2 Greedy Algorithm with Pairwise Collision Approximation

The true incremental reach $M(i)$ generated by choosing time slot i can be expressed as the raw impressions generated by i, less the overlap with the time slots that have already been chosen. However, as noted previously, the *true* overlap (and therefore the *true* reach) is prohibitively complex to calculate. With a similar spirit to the considerations of the linear-quadratic programming approach in Section 4, we therefore use pairwise collisions of time slot i with each already-chosen time slot as an approximation to compute the incremental reach $M(i)$ which would be produced by choosing slot i. Simpler approximations for reach can be found in Danaher [1991].

Let $\mathbf{P}_{i,i'}$ denote the proportion of viewers in the target demographic of network–time slot i' who also watch network–time slot i. With a slight modification of the data-driven approximation from Subsection 4.2, we estimate:

$$\mathbf{P}_{i,i'} = \frac{\sum_{v \in \mathcal{V}_i \cap \mathcal{V}_{i'}} d_v}{\sum_{v \in \mathcal{V}_{i'}} d_v}.$$

Here the greedy algorithm puts an order to the chosen slots, so we will use the impressions of the already-chosen slot i' as the reference (rather than the slot with the smaller number of impressions as in Subsection 4.2.) Then, using the idea of the pairwise collision approximation to the reach from Panaggio et al.

[2016], we can approximate the incremental reach $M(i)$ from selecting slot i at iteration step k as:

$$M_2(i; i_1, i_2, \ldots, i_{k-1}) \equiv S_i \prod_{j=1}^{k-1} (1 - \mathbf{P}_{i_j, i}),$$

where $\{i_1, i_2, \ldots, i_{k-1}\}$ are the previously chosen slots for placing an ad. At step $k = 1$ of course the greedy algorithm just choose the slot with the most impressions on the target demographic; the product is understood to be 1.

With this formulation, we can express $M(i_k, G)$ as

$$M(i_k, G) = G^{\text{imp}}_{i_k} \prod_{j=1}^{k-1} (1 - P_{i_j, i_k}).$$

Under the current setup of this algorithm, we simply stop when the choice of the next slot i causes the accumulated cost C to exceed the budget limit L. A possible improvement is to start searching, at this point, for the slot with the most incremental reach that is still cheap enough not to exceed the budget limit. However, this may result in populating $\mathbf{X}$ with many cheap time-slots, thereby becoming a possible target of a Saturday Night Live parody, which may or may not be desirable. The pseudocode for the greedy algorithm is summarized in Algorithm 1.

Algorithm 1 Greedy algorithm with only cost constraints

```
$\mathcal{S} \leftarrow \{1, \ldots, N^{(s)}\}$;   $C \leftarrow 0$;   $k \leftarrow 1$;   $\mathbf{X} \leftarrow \mathbf{0}$;
while $C \le L$ and $\mathcal{S} \neq \emptyset$ do
  Select $i_k \in \mathcal{S}$ that maximizes $\frac{S_{i_k} \prod_{j=1}^{k-1} (1 - \mathbf{P}_{i_j, i_k})}{c(i_k)}$
  if $C + c(i_k) \le L$ then
    $C \leftarrow C + c(i_k)$
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{i_k\}$
    $k \leftarrow k + 1$
    $\mathbf{X}_{i_k} \leftarrow 1$
  else
    break
  end if
end while
```

5.3 Greedy Algorithm with Final-State Constraints

One difficulty of a greedy algorithm is the accommodation of final-state constraints; by nature, a greedy algorithm is concerned only with the present, not with the future. As a specific example, our working group wrestled with the question of how to satisfy an equality constraint that 50% of the chosen slots must be between 3:00 AM and 6:00 AM.

We formulated and tested two alternatives. The **penalty method** is to include a term in the objective function that, at each iteration step, penalizes failure to satisfy the constraint with the incomplete schedule built to that point. The **partition method** divides, at each iteration step, the current set of available slots $\mathcal{S}$ into D_1 and D_2 as follows:

$$D_1 = \{i \in \mathcal{S} : i \text{ occurs between 3:00 AM and 6:00 AM}\},$$
$$D_2 = \{i \in \mathcal{S} : i \text{ occurs before 3:00 AM or after 6:00 AM}\}.$$

At iteration k, if the selected slots result in an undershooting of the target ratio, we restrict the choice in iteration $k + 1$ of Algorithm 1 so that $i_{k+1} \in D_1$; if iteration k results in an overshooting, we restrict the choice space so that $i_{k+1} \in D_2$.

5.4 Test Implementation

In our test implementation on the sample data set provided by clypd, slots i were defined using only time-of-day information, not day-of-week information, due to the limitations of the dataset. We used MATLAB to estimate $\mathbf{P}$ from historical Nielsen data, while S_i was obtained from a clypd forecast of the future impressions generated by airing an advertisement on network–time slot i. Our results, shown in Figure 2 and Table 1, indicate that the partition method requires more iterations but less runtime, and results in a higher reach. The difference in runtime can be attributed to the need to recompute the penalty for each state at each iteration of the algorithm.

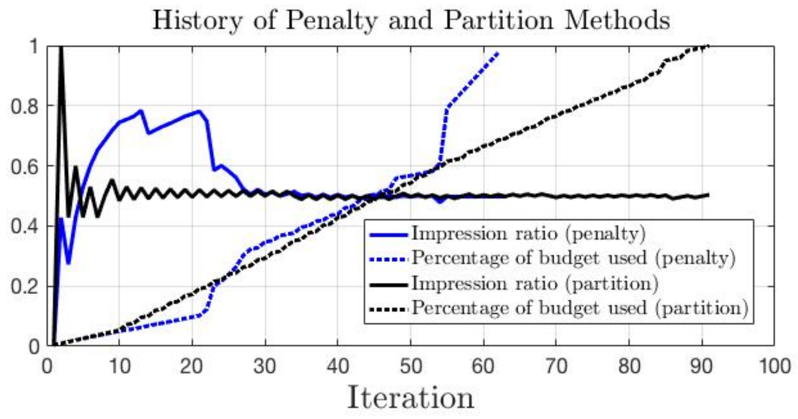

Figure 2: Iteration history of Penalty and Partition methods.

	Runtime (s)	Reach (persons)
Penalty	6.19	271,905
Partition	1.14	526,113

Table 1: Performance using target ratio of 50% and budget limit $L = 20,000$. These simulations were run on a MacBook Pro with a 2.9 GHz processor and 8 GB of memory.

6 Outlook and Future Directions

Three algorithmic approaches have been proposed to improve the reach to a desired demographic group of an advertising campaign schedule. The first modifies a linear programming approach through choosing the coefficients of the linear objective function with reference to a prior clustering analysis for similarity of viewership between potential advertising slots. The second employs a quadratic objective function with linear constraints, with the quadratic terms parameterized by estimates of viewership overlap between two slots. The third employs a greedy set coverage algorithm, which iteratively builds the schedule by optimizing the increase in estimated reach to the desired demographic with each additional scheduled slot. One challenge in the design of this third algorithm is the incorporation of many linear constraints. Our implementation on a test data set indicates that an approach based on restricting, at each step of the greedy algorithm, the candidate slots may be more efficient than representing the constraints through penalty terms in the objective function.

The presentation of the algorithms in this report has generally focused on impressions and reach with respect to one target demographic. Often advertising campaigns have both a primary target demographic as well as an "advanced target" which is more narrowly defined in terms of activities or interests. In the abstract, we can handle questions of reach and impressions on such advanced targets in completely parallel ways to simpler target demographics, so long as one can associated the Nielsen viewership panel to weights with respect to these advanced targets. The more substantial question is how to simultaneously improve reach for two different target groups, with the advanced target typically being a subset of the primary target demographic. Such multiobjective optimization, with constraints, typically proceeds by either placing weights of importance on each of the objectives (that is, how much more value is placed on reaching one more advanced target versus one more person from the primary target demographic) or by a splitting approach which seeks to optimize one objective function while possibly constraining the other objective function. A splitting approach for reach optimization might seek first to optimize the reach within the advanced target group, which, being a smaller subset, might allow the optimization procedure to find a solution faster. The more difficult question is how to take into account considerations of reach to the broader primary demographic while the reach is being optimized within the advanced targets. Reach is inherently nonlinear, so constraining the reach on the primary demographic to be somehow "good enough" while optimizing reach in the advanced targets would introduce nonlinear constraints that could be quite difficult to handle. A more feasible alternative might be constraining the number of *impressions* on the primary demographic while optimizing reach within advanced targets, since this would remain a linear constraint problem. Then one could take the solution as a basis for a subsequent optimization of reach within the primary demographic, while constraining the number of impressions within advanced targets to be some fraction of that achieved from the first optimization, and see if the solution is more satisfactory. The procedure could be iterated, but one clear deficiency is the loss of reach consideration for one of the two groups at each iteration step. Other heuristic alternatives could be to optimize reach of the advanced target until a certain fraction of the budget is spent or reach goal is met, then switch to optimizing reach of the primary target demographic. Such a two-step optimization approach as well as the choice of an objective function that gives certain relative value to reach within each of the two groups under consideration, are simple and attractive extensions of the methods developed in this report for a single target group.

All approaches we have discussed can be parameterized in terms of estimates for the pairwise overlap of viewership of potential advertising slots, meaning the probability that a viewer (from a given demographic) of one potential slot will also view a certain other slot. We have indicated in Section 4 how such pairwise overlap viewership can be estimated from historical Nielsen data in terms of the relative separation in time of the slots within and between days, their dayparts, and their genre overlaps. Such a pairwise approximation, described in more detail in the 2015 MPI report [Panaggio et al., 2016], does neglect potentially important sustained dependencies between viewership of different slots. One key example might be an episodic series, or news program, which, in an idealized sense, has a core viewership that watches every episode, and a completely random viewership that tunes into a given episode but has negligible probability to view any other episode in the near future. If this series has roughly constant viewership $\bar{S}$, with a fraction c being the core viewership, then a pairwise consideration would assign a probability c that a viewer of one episode will view a different episode. The reach of an advertisement placed on two different episodes would be $\bar{S}+(1-c)\bar{S}$, with the second term representing the number of new impressions made on the second episode due to the random component of the viewership. Now if we contemplate scheduling an advertisement in yet another episode, the number of new viewers will be estimated by a pairwise model as $\bar{S} - c\bar{S} - c\bar{S} = (1-2c)\bar{S}$, with the two subtracted terms representing the estimated overlap of viewers from the first slot and the second slot respectively. This underestimates the actual additional reach $(1-c)\bar{S}$ that would be obtained by the third slot under the idealized viewership model, because the pairwise approximation does not take into account sustained viewing patterns. The parameterization of our algorithms, therefore, could be improved through some incorporation of committed viewing patterns (of soap operas, news programs, sports events, late night shows, etc.) that are neglected by the pairwise reach approximation. This should be possible to estimate from historical Nielsen data.

Finally, we ponder whether the approaches we have developed here could be extended to achieve goals related to other properties of the exposure distribution, other than reach. For example, an advertiser might find it desirable to maximize the number of viewers who see an ad at least k times. Such exposure statistics are, first of all, even more complex mathematical functions of historical Nielsen data than reach (Eq. (2)), since they amount to more complex operations than simple unions of viewership lists. Computationally, they would be somewhat more expensive to compute, probably by iteratively building lists of viewers that have seen the ad exactly k' times. We already considered exact calculations of reach too impractical within an optimization framework, and in all cases resorted to approximations of reach based on the likelihood of a viewer to watch particular pairs of slots. In principle, we could still use this probabilistic model in connection with the greedy algorithm from Section 5 to estimate how many viewers would see the ad in a slot under consideration for the k'th time, though the approximations made in this model that only considers pairwise relations between slot viewership will likely lead to poor prediction quality for $k' > 2$. For considerations of high repeat viewership, surely some more sophisticated model of viewing patterns along the lines described in the previous paragraph would be indicated. The clustering and linear/quadratic programming approaches from Sections 3 and 4 are even more challenging to extend to more general exposure statistics because the simplicity of the objective functions make it difficult to tune to encourage a desired amount of moderate repeat viewership. The modification of the linear coefficients in Eq. (5) or the choice of the quadratic coefficients in Eq. (6) could rather naturally be used to minimize or maximize the frequency of viewings, but extending their use to other exposure statistics is an unclear endeavor. This all points to a fundamental conceptual problem with trying to optimize more complex exposure statistics than reach (or frequency) – how can a potential schedule be given a value that reflects how well it meets some exposure distribution goal, in a manner that is much less expensive to compute than its exact evaluation from historical or simulated future Nielsen viewership data? Perhaps this would be a relevant problem for a future MPI workshop. . .

References

Chandra Chekuri. Approximation algorithms, Fall 2013. URL `https://courses.engr.illinois.edu/cs583/fa2013/`.

Peter J. Danaher. Optimizing response functions of media exposure distributions. *J. Opl Res. Soc.*, 42(7): 537–542, 1991.

Marco A. Montes de Oca. clypd scheduler: Master document. Technical report, clypd, June 20 2017.

Peter Lusztig and Bernhard Schwab. A note on the application of linear programming to capital budgeting. *Journal of Financial and Quantitative Analysis*, 3(4):427–431, 1968.

M. J Panaggio, P.-W. Fok, G. S Bhatt, S. Burhoe, M. Capps, C. J Edholm, F. El Moustaid, T. Emerson, S.-L. Estock, N. Gold, R. Halabi, M. Houser, P. R Kramer, H.-W. Lee, Q. Li, W. Li, D. Lu, Y. Qian, L. F Rossi, D. Shutt, V. Chuqiao Yang, and Y. Zhou. Prediction and Optimal Scheduling of Advertisements in Linear Television. *ArXiv e-prints*, August 2016.

Arthur Szlam and Xavier Bresson. A total variation-based graph clustering algorithm for Cheeger ratio cuts. *UCLA CAM Report*, pages 09–68, 2009.