TV Advertisement Scheduling by Learning Expert Intentions

Yasuhisa Suzuki
NEC Corporation
y-suzuki@jy.jp.nec.com

Wemer M. Wee
NEC Corporation
w-wee@bk.jp.nec.com

Itaru Nishioka
NEC Corporation
i-nishioka@cb.jp.nec.com

ABSTRACT

This paper considers the automation of a typical complex advertisement scheduling system in broadcast television (TV) networks. Compared to traditional TV advertisement scheduling, we consider the case where not all requests for advertising slots are known at the same time, and time-consuming negotiations related to balancing the TV network's and advertisers' priorities have to be minimized. Although there are existing works that automatically provide schedules using mathematical optimization, the applicability of these techniques to our problem is limited due to the cumbersome formulations necessary for handling vague conditions and aesthetic domain-specific rules necessary for advertisers' satisfaction.

To automate the system, we propose a data-driven approach that uses intention learning on top of mathematical optimization and clustering to imitate the decision-making process of scheduling experts. The scheduling of TV ads is automated via mathematical programming, and the expert objectives and constraints are learned from historical demonstrations using inverse optimization. The clustering of TV ads and the learning of associated intentions are used to deal with the cold start problem related to ad requests from new companies or products. Our proposed system is validated on actual dataset from a Japanese TV network. Experiments show that our system can more closely reproduce the experts' schedules compared to standard optimization approaches, demonstrating the potential of our work in reducing personnel costs and improving advertisers' experience. Based on its promising results, our proposed system is being prepared for commercial deployment.

CCS CONCEPTS

• **Applied computing** → **Marketing**; • **Computing methodologies** → **Learning from demonstrations**.

KEYWORDS

Scheduling, Operations Research, Inverse Optimization, Clustering

ACM Reference Format:
Yasuhisa Suzuki, Wemer M. Wee, and Itaru Nishioka. 2019. TV Advertisement Scheduling by Learning Expert Intentions. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330768

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330768

1 INTRODUCTION

Advertising in broadcast television (TV) is one of the most effective marketing strategies for promoting goods, services or even company brand image to a wide range of potential customers. In Japan, TV advertisements (ads) dominate and make up around 30 percent (2 trillion yen) of all advertising media in 2018 [15]. In recent years however, online ads have become increasingly popular, which threatens the growth of advertisement (ad) spending on TV. Due to this intense competition, TV networks are determined to improve their operational efficiency, which makes them inclined to undergo automation of traditional processes and employ techniques that leverage their own data.

Unlike highly automated online ads, traditional TV ad scheduling is still done manually, and there are several unique obstacles to consider before a high level of automation can be achieved. One is that compared to online ads in which the assignment is a matching problem between specific individuals going online and goods they may prefer, TV ad scheduling deals with a very limited number of slots to which the most suitable TV ads should be assigned. This problem is more complex as conflicts arising from different advertisers' requirements are needed to be resolved in order to fill up all available slots and maximize the TV network's revenue.

Another issue is that there is a large number of factors in TV ad scheduling that are not easy to model mathematically or be associated to simple attributes. In online ads, the customer type can be observed and the effect of the ads can be measured from their reactions. In contrast, TV ad scheduling still heavily relies on human intuition as effects depend not only on time, but also on the program and ad contents, genre, actors, and other brands that will be shown. Examples of objectives or constraints coming from human insights are: (i) a company's ad should not be placed in a particular program or near another ad as the company's brand image will be affected, (ii) ads of competing products should be separated and not located at the same time of day because their commercial effects are cut in half. These constraints are carefully considered by TV network experts, and this makes it difficult to apply existing techniques similar to those used for online ads.

There already are several efforts made for automating TV ad scheduling, notably in the operations research area using mathematical programming techniques [5–7]. However, these efforts have been successful in only a limited number of TV networks so far, as it is very costly and needs several customizations in order to fit different TV network operation styles. This is in contrast to the more common solution that can be found for online ads. Also, compared to traditional settings, the following has not yet been fully considered: first is that not all TV ad requests for slots are known at the same time, and second, time-consuming back-and-forth negotiations with a customer's advertising agency should be reduced. These conditions are very common, and although having an auction-style system for TV ad scheduling can be advantageous

to the TV network, an *on-demand* style wherein advertisers receive schedules for their ad requests quickly can do a lot in terms of improving advertisers' satisfaction and experience. Due to these reasons, most TV networks have not yet achieved automation of their ad scheduling systems.

To address the above problems and provide a more flexible and data-driven approach to the automation of TV ad scheduling systems, this paper proposes the use of *intention learning* on top of mathematical optimization and clustering. Our proposed system can imitate the decision-making process that TV clerks or experts perform to create schedules that satisfy their advertisers. Intention learning approaches have been gaining increasing popularity recently, and many advances have been made in what are called *inverse reinforcement learning* (IRL) [20] and *inverse optimization* (IO) [2] techniques. In this work we consider the use of IO as the scheduling operations can be more simply assumed to be based on solving an optimization problem, instead of employing a Markov decision process as in IRL. Like other intention learning approaches, IO enables us to learn objective functions and constraints from historical demonstrations of human experts, without requiring them to explicitly state their intentions, rules or preferences. The insights from the experts learned from data are then used in a mathematical program to automatically generate schedules that can improve both the TV network's revenue and advertisers' satisfaction. For the application we are considering, we propose the use of a probabilistic formulation of inverse optimization as in the IRL technique in [28], which is based on the principle of maximum entropy [16].

In addition, we also deal with the associated cold start problem, where a TV ad request from a new company or product comes and clearly no historical information is available. This is resolved using a hierarchical clustering method. Once meaningful clusters are learned, we then apply IO to recover intentions that apply to entire product categories. Such representative intentions are then used in helping to choose a suitable objective function in the mathematical program. The proposed solution for automating TV ad scheduling is thus a synergy between inverse optimization, mathematical programming and clustering techniques, and we apply it to a real dataset from a Japanese TV network and show the potential of the system in improving TV clerks' productivity.

Our main contributions are summarized as follows:

(1) We develop an inverse optimization method which learns the expert objectives and constraints for use in a mathematical program we formulate for TV ad scheduling.
(2) We solve an associated cold start problem by applying a suitable clustering method and recovering intentions for the learned ad clusters.
(3) We demonstrate a high level of scheduling automation and imitation of experts using actual data from a TV network.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the problem in detail by referring to the workflow in an actual Japanese TV network. In Section 4, we propose a probabilistic inverse optimization technique for learning experts' objective functions and constraints. We also describe how the cold start problem is addressed. Section 5 shows the experimental results of applying our proposed system on the actual dataset. Finally, we conclude our discussion in Section 6.

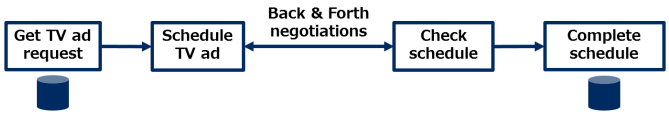

Figure 1: An overview of the TV ad scheduling workflow. The two deep blue cylinders represent input data in the learning phase.

2 RELATED WORK

The earliest works that deal with the problem of TV ad scheduling are described in [5–7], with the developed systems having been deployed at NBC in the US. These works resolve conflicts arising from advertisers' requests using mathematical programming techniques and automatically generate ad schedules. Similar works include [21, 23], which present methods for generating schedules that maximize advertising revenue.

In our problem however, employing such techniques is not sufficient, as we want to deal with negotiations that involve many subtleties related to the relationships between different advertisers and the TV network. Moreover, we assume that ad requests arrive, and corresponding schedules are created, one at a time. These constraints require a different approach to ad scheduling.

Inverse optimization [2] is a technique for learning from demonstrations that has been getting popular recently. Due to their potential in improving decision-making tasks using expert demonstrations, IO techniques have been developed in many different situations, such as in online settings [4, 9], imperfect observations [3, 11], multi-objective settings [8, 10] and robust optimization [14].

Another related group of techniques is based on IRL [1, 20]. Compared to the IO setting, many IRL techniques [24, 28] are based on a probabilistic setting and can be treated in a maximum likelihood or maximum a posteriori estimation framework. An advantage of a probabilistic model is that suboptimal behavior can be explained as noise. In particular, the MaxEnt IRL approach [27, 28] has been shown to be a state-of-the-art approach to recovering an expert's objectives given past demonstrations or trajectories. We thus adopt the same approach to develop a probabilistic inverse optimization technique.

3 PROBLEM DESCRIPTION

In this section we describe the problem in detail and give a formulation of TV ad scheduling as a mathematical optimization problem.

3.1 Scheduling Workflow

An overview of the TV ad scheduling workflow in a TV network in Japan is shown in Figure 1. A brief description is given as follows. First, an advertiser sends ad request data to a TV clerk. Second, based on the ad request, a TV clerk has to quickly create a schedule that meets as many of the advertiser's requirements as possible, but at the same time, a high importance to the interests of the TV network is taken into account. Third, a version of the schedule is shown to the advertiser, and several negotiations occur wherein both sides try to find a mutually acceptable schedule. This goes under multiple iterations and is the most difficult and time-consuming

part of the workflow. The operation is finally completed once a suitable schedule is found.

The workflow described is followed by about 15 clerks that execute the tasks simultaneously, and in each month, they create about 750 schedules in total. For such a large number of TV ads, the repetition of exploratory trial-and-error is a cumbersome and time-consuming job. However, the automation of this particular workflow is hindered by three main issues.

The first issue is related to how a schedule that satisfies certain requirements can be automatically created. This can be solved using mathematical optimization. Although there are some previous works related to this, we have to formulate our own optimization problem to create schedules using features or constraints that are relevant to this particular situation.

The second issue deals with how negotiations between the TV network and an advertiser can be minimized or avoided. As the negotiations are done to find a schedule acceptable to both sides, this can be considered to be a process of tuning importance weights of an optimization problem. This has not been considered before in advertisement scheduling, and we solve this problem via inverse optimization. Once the parameters or intentions for a particular brand or product are recovered from historical data, they can be used in the formulated optimization problem to automatically generate a suitable schedule.

The third issue relates to a cold start problem, which considers how a schedule can be created given an ad request from a new company or product. To solve this problem, we perform clustering on the past ads to build groups of ads that exhibit similar schedules or broadcasting strategies. Inverse optimization can then be applied to recover intentions present in the ad clusters. The intentions are then used in automatically generating schedules for new ads.

In summary, we consider the following for automating our specific workflow:

- automatic scheduling with constraints via mathematical optimization,
- learning intentions or data-driven tuning of a mathematical program via inverse optimization, and
- solving the cold start problem by clustering and learning representative intentions.

By solving the above issues, we develop a system which automatically creates schedules similar to what TV clerks intend, with no heavy modifications necessary and reducing the cumbersome part of TV ad scheduling.

3.2 TV Ad Scheduling Formulation

We first consider the problem of ad scheduling. We assume that TV clerks solve mathematical optimization problems to come up with reasonable schedules of the ads during their contract periods.

The ad request data received by TV clerks contain the advertiser's preferences about how they want the ad to be broadcasted, by indicating, for example, their desired contract period, the duration of the ad, time constraints, and their required total gross rating point (GRP) [13], which gives a measure of the size of an advertising campaign based on *impressions*.

The total GRP requirement is the minimum which must be acquired during the contract period. Clearly, it is possible for an ad

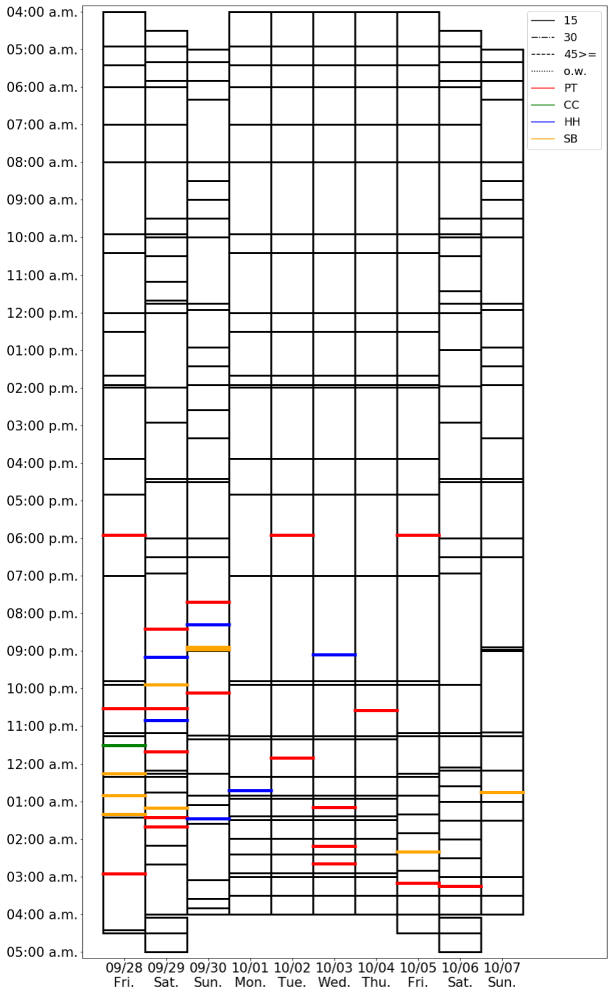

Figure 2: An example schedule for a beer TV ad, with contract period 2018/9/28-2018/10/7. The rectangle boxes are TV programs and all of the colored horizontal bars indicate the places where the TV ad is broadcasted. The line style represents the duration of the TV ad (in this figure, only 15 seconds), and the color shows the slot type, which may indicate that the ad is broadcasted between two different programs (SB), within one program contents (PT), or others (CC and HH). The ads for alcoholic drinks in Japan are not broadcasted between 5:00 a.m. and 6:00 p.m. because of self-restraint [18], so the ads only appear in night slots.

to acquire extra GRPs, but it means that a TV clerk has created an inefficient schedule for the TV network since it relates to missed opportunities to put in other ads.

An example of an ad schedule is shown in Figure 2. Note that the timing of the ad requests depends entirely on the advertisers, so they are usually sent one at a time and schedules have to be created for individual ads. The scheduling is the assignment of ads into multiple slots belonging to different TV programs. The orders of different TV ads in each slot are strongly considered only if there is

enough information that can be used to relate it to the program and other ads. Each slot has a duration time alloted for broadcasting ads, and the value of a slot is affected based on its *"slot type,"* which may refer to whether it is between two different programs or between program contents.

In many cases, the remaining slots may not have enough space for new TV ads as duration times or GRPs have already been taken by earlier requested ads, so the clerk moves some earlier ads to other places after negotiations with other clerks. Moreover, once a TV clerk has created a working schedule, the negotiation process with the advertiser starts. Note that this also involves negotiations with other clerks for checking the consistency of the entire schedule. The schedule usually undergoes several modifications until a version is accepted by the advertiser.

Ad requests may include the target audience, the names of preferred or unacceptable programs or program genres, and so on. However, these information are not available in standard form, and they are very difficult to process as they are written using unique terminology and not based on standard abbreviations.

Thus, when scheduling an ad, a TV clerk must consider

- the satisfaction of the advertiser's requirements,
- the remaining seconds of each slot,
- the balance between the TV ad and other ads, and
- possible adjustment with other clerks.

Based on the above description, we are now ready to formulate our mathematical program. In the overall scheduling problem, the tradeoff between all the advertisers' intentions and the TV network's objectives needs to be considered. On one hand, advertisers want to broadcast their ads in a way that maximizes cost performance, that is, to maximize GRP while satisfying imposed constraints for keeping their brand image. On the other hand, the TV network wants to satisfy the advertisers' requirements with the smallest number of slots possible to reserve some space for newer ad requests, so that revenue can be increased. Thus, the objective function for the overall problem can be represented in the form

$$J = w_{tv} J_{tv} + \sum_{adv} w_{adv} J_{adv}, \tag{1}$$

where w_{tv} and w_{adv} are weighting parameters which represent the relative importance between the terms important to the TV network and the advertisers, respectively.

However, in the actual workflow, each TV ad request arrives one at a time, and the TV clerks need to create the schedules as quickly as possible. To fit their workflow, we formulate an optimization problem for creating a schedule for one TV ad at a time. In this case, we can consider an objective function J_i for scheduling TV ad i as composed of two major parts, one for the TV network and one for the advertiser of the particular ad. The objective function J_i for scheduling a TV ad i can then be described as

$$J_i = w_{i,tv} J_{i,tv} + w_{i,adv} J_{i,adv}. \tag{2}$$

Now let $x_{i,j}$ be the decision variable, where $x_{i,j} = 1$ if TV ad i is put into slot j, and 0 otherwise. The whole schedule can then be represented by a column vector $\boldsymbol{x}_i \in \{0,1\}^{n_i}$, which contains all $x_{i,j}$'s with n_i being the total number of slots in the contract period of TV ad i. Constraints are represented simply as $\boldsymbol{x}_i \in X(u_i)$, where u_i is the parameter that depends on the advertisers' requirements.

For simplicity, we assume that each objective function part is an inner product of a parameter θ and a feature vector of the schedule of the ad $\phi(\boldsymbol{x}_i)$, so by combining the parameters w and θ, we can write the objective function J_i in the form

$$J_i = \theta_{i,tv}^{\mathsf{T}} \phi_{tv}(\boldsymbol{x}_i) + \theta_{i,adv}^{\mathsf{T}} \phi_{adv}(\boldsymbol{x}_i). \tag{3}$$

One can then combine the parameters and features and write the optimization problem used for scheduling a TV ad i in the form

$$\max_{\boldsymbol{x}_i \in \{0,1\}^{n_i}} \quad \theta_i^{\mathsf{T}} \phi(\boldsymbol{x}_i) \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{x}_i \in X(u_i). \tag{5}$$

Features of the schedule which are important to the two sides are used. For example, for the TV network, the total remaining seconds of the slots is necessary as if its value is small, then there is no more space for other TV ads. On the other hand, for the advertiser, the GRPs acquired at each time is considered. The features can also be chosen in a way that the resulting optimization problem will be at most quadratic to prevent expensive computations.

As a result, the following features are extracted from the schedules and used to represent the TV network's features ϕ_{tv}:

- The total remaining seconds of all slots after scheduling.
- The total remaining seconds of only the most expensive slots after scheduling.

On the other hand, the following are considered as relevant to the advertiser's features ϕ_{adv}:

- Total GRPs acquired at each time slot.
- Total GRPs acquired during weekdays.
- Total GRPs acquired for each slot type.
- Total GRPs acquired for each time rank, which reflects the price rank of the slots.

The main constraints we use are listed below, all of which can be represented in linear form:

- The remaining seconds for each slot after scheduling should be nonnegative.
- The total acquired GRPs must not be less than required.
- Time constraints of TV ads, if available, should be satisfied.

Note that if we allow situations where a TV ad can be placed in one slot many times, we need to consider the relationship between the different TV ads to avoid concentration into the best slot. However, the problem becomes highly nonlinear and difficult to solve. For simplicity, we add a reasonable constraint in which the number of ads in the same slot is at most one.

4 AUTOMATED TV AD SCHEDULING

As is done traditionally, ad schedules can be obtained automatically by solving optimization problems such as (4) in the previous section. However, using such an approach by itself may not be able to capture different nuances that exist considering the relations between the TV network and the different advertisers or brands. Our fundamental idea is to capture those intentions by applying inverse optimization, which can give us the clerk or expert parameters θ_i in (4) by learning from their previous demonstrations. This allows us to generate schedules automatically that are close to what are intended by using the learned parameters in the optimization problem (4).

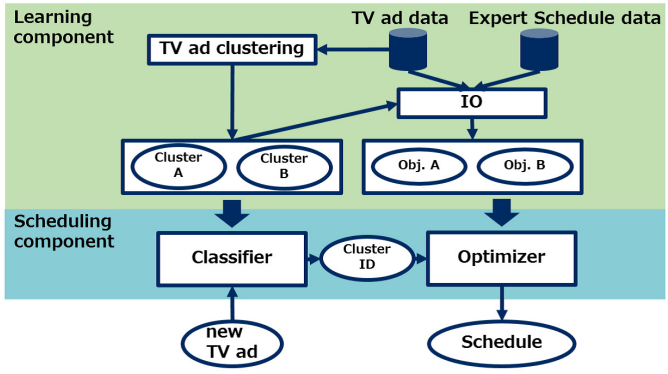

Figure 3: A rough sketch of the entire system configuration. The learning and scheduling components replace the back-and-forth negotiations shown in Figure 1.

To fully deploy the system, we have an additional issue related to the cold start problem, in which we do not have any learned objective for ads from new products or brands. To overcome this problem, we develop an approach that applies a clustering method on the past TV ads, and then learn objective functions for each cluster. For the new TV ads, once they are assigned to a suitable cluster, we can then create expert schedules using the learned objectives. An additional benefit to this approach is that the learned objective functions can avoid overfitting to the past data because of the variety of schedules present in the learned clusters.

4.1 Solution Overview

Our system is divided into two parts, a learning component and a scheduling component. Figure 3 shows a rough sketch of the configuration of the entire system.

The learning component is for preprocessing and has two main tasks, clustering the TV ads and applying inverse optimization to each cluster. We first apply a suitable clustering method to past TV ads to get clusters of schedules with similar broadcasting policies. We then apply IO to each cluster to learn the objective functions that represent each cluster from the expert schedule data of all cluster members. Note that we assume that TV ads with similar broadcasting policies have similar objective functions. In this component, the TV ad request data and expert schedules are the inputs, and the TV ad clusters and the cluster objective functions are the outputs.

The scheduling component creates schedules for input TV ads. Once this component receives TV ad data, it chooses a suitable cluster based on its broadcasting policy and characteristics. After cluster assignment, an optimization problem using the learned objective for the cluster is then solved to get a suitable schedule for the TV ad.

4.2 TV Ad Clustering

Ideally, we want to perform clustering of broadcasting policies using informative TV ad attributes such as program preference or target audience. In our case however, we are not able to use such information as they are either written in free format, difficult to be processed, or simply unavailable. We therefore applied clustering on the features extracted from the historical expert schedules. Assuming that the program genres are determined by broadcasting time and day of week, it is possible to satisfy the related constraints by checking the time and day related features. However, in the scheduling phase, it is not possible to choose a suitable cluster automatically for new TV ads as there are no available schedules that can be used as training data. So in this case we ask the TV clerks to choose the clusters manually by showing them information or features related to different clusters. This is a temporary problem, as the clusters can be updated once the relevant information becomes available. By choosing a suitable cluster, the clerks can then automatically generate the schedules.

For clustering, we choose the agglomerative hierarchical clustering method [25]. This is because the broadcasting policies of TV ads are created in a hierarchical manner, for example, the target audience may be determined first based on characteristics of the product, and then the preferred programs can be selected based on the preference of the target audience. The agglomerative hierarchical clustering algorithm is described as Algorithm 1. For this algorithm, we need to make two choices, a linkage criterion which is a distance function between two clusters, and a distance function between two schedule features.

Algorithm 1 Agglomerative Hierarchical Clustering

Input: Data points $D = \{x_1, ..., x_n\}$.

1: Create n disjoint clusters so that all of the clusters have only one member.
2: **while** the number of clusters is not equal to one **do**
3: Compute distance between all pairs of different clusters.
4: Combine two clusters which have the minimum distance among all the pairs.
5: **end while**

For the linkage criterion, we apply the complete linkage method as it tends to create spherical clusters which have approximately equal diameters [12], allowing us to get clusters that each have a reasonable number of members. In the criterion, updating the distance between two clusters is based on the following: when combining clusters c_a and c_b to get a new cluster c_{a+b}, the distance d between the new cluster c_{a+b} and another cluster c_k is given by

$$d\left(c_k, c_{a+b}\right) = \max\left\{d\left(c_k, c_a\right), d\left(c_k, c_b\right)\right\}. \tag{6}$$

We apply the clustering method using features ϕ_{adv} relevant to the advertiser, which are extracted from the expert schedules. An example of this kind of feature can be the GRPs acquired on each weekday. The length of the feature vector does not have any particular importance. To illustrate, when we have two different TV ads with total GRP requirements 100 and 200, and the respective acquired GRPs on Monday and Tuesday are 70% and 30% in both schedules, we assume that these two have similar broadcasting policies. To identify such TV ads, we apply the cosine distance to measure the distance between feature vectors of their schedules.

The distance between two schedules is given by

$$d\left(\phi_{adv}\left(\boldsymbol{x}_i\right), \phi_{adv}\left(\boldsymbol{x}_j\right)\right) \tag{7}$$

$$= 1 - \cos\left(\phi_{adv}\left(\boldsymbol{x}_i\right), \phi_{adv}\left(\boldsymbol{x}_j\right)\right) \tag{8}$$

$$= 1 - \frac{\phi_{adv}\left(\boldsymbol{x}_i\right)^{\mathsf{T}} \phi_{adv}\left(\boldsymbol{x}_j\right)}{\|\phi_{adv}\left(\boldsymbol{x}_i\right)\|_2 \|\phi_{adv}\left(\boldsymbol{x}_j\right)\|_2}, \tag{9}$$

where $\|\boldsymbol{x}\|_2$ is the ℓ_2-norm of the vector $\boldsymbol{x}$.

4.3 Inverse Optimization

After TV ad clustering, we perform inverse optimization to learn objective functions for each cluster. Compared to earlier IO techniques, we consider a probabilistic IO approach based on the principle of maximum entropy [16], which we refer to as MaxEnt IO. A description of the algorithm is given below.

We treat inverse optimization as a maximum likelihood estimation problem. Thus, the solution to the inverse optimization problem is obtained by solving

$$\max_{\theta_c \in \mathbb{R}^{n_\phi}} p\left(X_c|\theta_c\right), \tag{10}$$

where

- θ_c is the expert parameter of cluster c,
- n_ϕ is the dimension of the range of the feature map ϕ, and
- $X_c = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$ is the set of expert schedules of all member ads in cluster c.

For tractability, we assume that the schedules in X_c are independent, so we can decompose the likelihood as

$$p\left(X_c|\theta_c\right) = \prod_{i=1}^{m} p\left(\boldsymbol{x}_i|\theta_c\right). \tag{11}$$

We can thus write (10) as

$$\max_{\theta_c \in \mathbb{R}^{n_\phi}} L\left(\theta_c\right), \tag{12}$$

where

$$L\left(\theta_c\right) = \sum_{i=1}^{m} \log p\left(\boldsymbol{x}_i|\theta_c\right). \tag{13}$$

To maximize the above we can then use gradient ascent [26]:

$$\theta_c^{\mathbf{new}} = \theta_c^{\mathbf{old}} + \alpha \nabla L\left(\theta_c^{\mathbf{old}}\right) \tag{14a}$$

$$= \theta_c^{\mathbf{old}} + \alpha \sum_{i=1}^{m} \nabla \log p\left(\boldsymbol{x}_i|\theta_c^{\mathbf{old}}\right), \tag{14b}$$

where α is the step size.

For the likelihood function $p\left(\boldsymbol{x}_i|\theta_c\right)$, we use the following distribution which results from entropy maximization as discussed in the case of MaxEnt IRL [28]:

$$p\left(\boldsymbol{x}_i|\theta_c\right) = \frac{1}{Z(\theta_c)} \exp\left(\theta_c^{\mathsf{T}} \phi\left(\boldsymbol{x}_i\right)\right), \tag{15}$$

where $Z(\theta_c) = \sum_{\boldsymbol{x}_i' \in X(u_i)} \exp\left(\theta_c^{\mathsf{T}} \phi\left(\boldsymbol{x}_i'\right)\right)$ is called the partition function. Using the likelihood function above, the gradient is then given by

$$\nabla \log p\left(\boldsymbol{x}_i|\theta_c\right) = \phi\left(\boldsymbol{x}_i\right) - \mathbb{E}_{p\left(\boldsymbol{x}_i'|\theta_c\right)}\left[\phi\left(\boldsymbol{x}_i'\right)\right], \tag{16}$$

where the first term is the feature vector from the expert schedule, and the second term is the expected value of the features of the generated schedules based on the current estimated parameter. To get the value of the second term, we need to solve the forward optimization problem (4) and (5). As in [28], the value of the second term can be approximated using sample-based techniques. In this case in particular, the expected feature value is approximated by the feature value of the generated schedule given the conditions for ad i. We thus use the following approximation to the gradient:

$$\nabla \log p\left(\boldsymbol{x}_i|\theta_c\right) \approx \phi\left(\boldsymbol{x}_i\right) - \phi\left(\boldsymbol{x}_i'\right). \tag{17}$$

As a result, by substituting (17) for (14b), the parameter update equation is given by

$$\theta_c^{\mathbf{new}} = \theta_c^{\mathbf{old}} + \alpha \sum_{i=1}^{m} \left(\phi\left(\boldsymbol{x}_i\right) - \phi\left(\boldsymbol{x}_i'\right)\right). \tag{18}$$

The MaxEnt IO algorithm is summarized as Algorithm 2. We also summarize the components of the TV ad scheduling system in Algorithms 3 and 4.

Algorithm 2 MaxEnt IO for cluster c

Input: Expert schedules $X_c = \{\boldsymbol{x}_i\}_{i=1}^m$ of all ads in cluster c.
Output: The expert parameter vector $\hat{\theta}_c$.

1: Set the number of parameter update steps T and step size α.
2: Initialize the parameter $\hat{\theta}_c^{(1)}$.
3: **for** $t = 1, ..., T$ **do**
4: $\quad X_c^{(t)} \leftarrow$ Solutions of the forward problems (4) and (5) under the parameter $\hat{\theta}_c^{(t)}$ for all ads in cluster c.
5: $\quad \nabla L\left(\hat{\theta}_c^{(t)}\right) \leftarrow \sum_{i=1}^m \left(\phi\left(\boldsymbol{x}_i\right) - \phi\left(\boldsymbol{x}_i'\right)\right)$.
6: $\quad \hat{\theta}_c^{(t+1)} \leftarrow \hat{\theta}_c^{(t)} + \alpha \nabla L\left(\hat{\theta}_c^{(t)}\right)$.
7: **end for**
8: $\hat{\theta}_c \leftarrow \hat{\theta}_c^{(T+1)}$.

Algorithm 3 Learning Component

Input: All past ads and their expert schedules.
Output: Ad clusters $C = \{c_1, ..., c_n\}$ and expert parameters $\left\{\hat{\theta}_{c_1}, ..., \hat{\theta}_{c_n}\right\}$.

1: $C \leftarrow$ Ad clusters as obtained from the clustering method in Algorithm 1.
2: **for** $k = 1, ..., n$ **do**
3: $\quad \hat{\theta}_{c_k} \leftarrow$ Expert parameters for cluster c_k as the output of MaxEnt IO in Algorithm 2.
4: **end for**

5 EXPERIMENTS

In this section we check the effectiveness of our proposed scheduling system using a real dataset from a Japanese TV network. The target ads we use have contract periods belonging to the duration 2018/07-2018/12. An order to the ads is given based on the timestamps they were submitted to the advertisers, and we create schedules using that order one at a time. We then divide the time

Algorithm 4 Scheduling Component

Input: Ad clusters C, objective parameters $\left\{\hat{\theta}_{c_1}, ..., \hat{\theta}_{c_n}\right\}$, and new TV ad i.
Output: Schedule $\boldsymbol{x}_i$ of the new TV ad i.
1: $c_k \leftarrow$ Choice of most suitable cluster for the new TV ad i from cluster set C.
2: $\boldsymbol{x}_i \leftarrow$ Solution of the forward problem (4) and (5) based on the expert parameter $\hat{\theta}_{c_k}$ of cluster c_k.

Table 1: Some TV ads belonging to clusters chosen in Figure 4. Each member ad is represented using its product category.

Cluster A	Cluster B	Cluster C	Cluster D
Soda 1	Detergent 1	Beer 1	Beauty salon
Soda 2	Online supermarket	Beer 2	Camera retailer
Soda 3	Detergent 2	Beer 3	Truck
Soda 4	Consumer goods	Beer 4	Tourism promotion
Soda 5	Detergent 3	Sake	Smoking manner

series data into training and validation data sets, with 70% (about 2,300 ads) for training and the remaining 30% (around 1,000 ads) for validation. The training data is used in the learning component and the validation data is for testing the ability of the entire system.

5.1 Insights from Learning Component

We first describe the results of the learning component, which is comprised of TV ad clustering, IO and cluster assignment.

5.1.1 TV Ad Clustering. We extracted the advertiser features ϕ_{adv} from each expert schedule, and applied the agglomerative hierarchical clustering method with the complete linkage criterion. To get clusters from the result, we need to choose a threshold value. If the value is too large, the clusters can be too abstract to get meaningful objective functions. On the other hand, if the threshold value is too small, manually choosing the most suitable cluster for the new ad in the scheduling phase can be too difficult to the TV clerks because of the large number of choices. By considering this tradeoff, several values have been analyzed and the threshold is chosen to be 0.3, which resulted in 64 clusters. The piechart in Figure 4 shows the relative sizes of the clusters over all available TV ads.

Due to the complete linkage method, spherical clusters with approximate equal diameters are created, so we can approximate the centers with the average points among overall feature points in each cluster. To check the reasonability of the clustering result, we pick some representatives from the clusters in order of their closeness to the approximate centers. Table 1 shows some of the representative members.

In Figure 4, we can see that some clusters have many members while other clusters are relatively small. Large clusters are expected to be formed as there are many TV ads for which the target audience covers a wide spectrum, which means that they do not have a very strong or specific broadcasting strategy. As an example, the largest cluster, cluster D in Figure 4 and Table 1, has camera and tourism ads as members, and they are favored by all ages and considered popular hobbies. The large clusters have these TV ads that appeal

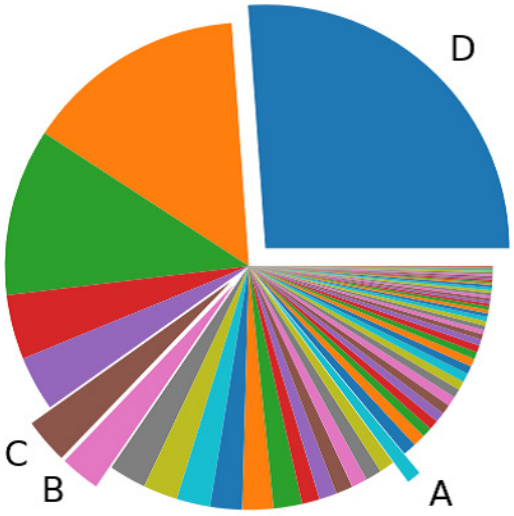

Figure 4: The piechart represents the relationship among the sizes of the clusters. The highlighted clusters correspond to those used in Table 1.

to a wide age range, so scattered placements are preferred to make a large variety of people familiar with the products, and thus the schedules do not have any very remarkable features.

There are also clusters having an intermediate number of member ads, each having a more targeted advertising approach. For example, consider clusters B and C in Figure 4, which have the kinds of products as shown in Table 1. Cluster B can be interpreted as an ad cluster targeting homemakers. These ads are broadcasted in the daytime as many homemakers and retired people watch TV shows during that time and other people are in offices or schools. On the other hand, cluster C is comprised of alcoholic drink ads. Alcoholic ads are not broadcasted in the daytime due to some self-restraint by beverage companies, and this is considered a very remarkable feature for this cluster.

Finally, the clusters which have a very small number of members can be divided into two types: the outliers and the characteristic advertisers. The latter specifically means that such clusters have only one kind of product of the same company, so they apply their original or unique policy for broadcasting their ads, as in cluster A in Table 1. This example cluster has about 20 members, and all of them are ads of the same soda drink.

One sees from the above that for TV clerks to satisfy as many requirements as possible, the schedules of the ads in the small and intermediate clusters can be created first, and the ones for the large clusters can then be created using the remaining slots. As a result, we conclude that member ads of the small or intermediate-sized clusters can be prioritized more than the ones belonging to large clusters.

5.1.2 MaxEnt IO. A learning curve from MaxEnt IO that describes the evolution of the difference between the feature vectors extracted from the expert and generated schedules is shown in Figure 5. The distance increased suddenly in the first update due to the choice

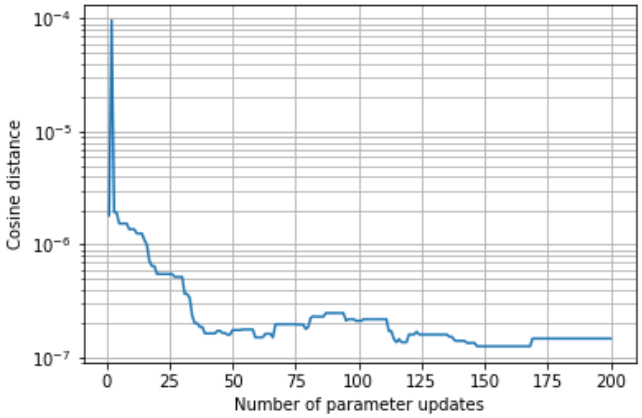

Figure 5: The learning curve of a house cleaning tool ad in Cluster B. The vertical axis represents the cosine distance (in log scale) between the features of the expert schedule and the generated schedule.

of initial parameters, which only gives importance to the priorities of the TV network. Observe however that after a few parameter updates, the difference is significantly decreased and reaches a reasonably low value, meaning that the expert and generated schedules are very similar in the sense of the chosen features.

On the other hand, to visualize or interpret the learned objective function, we show in Figure 6 a heatmap, which represents the learned preferences for each slot, as the objective function is just summing up the individual slot scores. It can be seen that, for the house cleaning product, the scores are high for the 2 p.m. to 6 p.m. slot. This is because during this time, only homemakers and retired people are mostly available, and programs which appeal to them are broadcasted. Note that the scores are also fairly high in the morning, as many people have breakfast with their family while watching TV. The interpretations derived from the learned objectives can be seen to match expected broadcasting strategies.

5.1.3 Cluster Assignment. In this experiment, we attempt to do automatic assignment to check the results of scheduling. We trained a classifier for choosing the clusters automatically by using the result of agglomerative hierarchical clustering as training data. Because we have the expert schedules for validation data, we can use them as the input to the classifier, so we can choose the cluster automatically. We use the K-Nearest Neighbors algorithm for classification. After cross-validation with 10 splits, the hyperparameter $K = 1$ needs to be chosen to achieve highest accuracy, which is about 72%.

This basic test shows that given the currently available data, training a good classifier is difficult, and one should consider training a model once informative features like genre and target audience are obtained. Thus in the real system, we may for now ask the TV clerks to choose a cluster by themselves. This is done by showing valuable information such as the schedule heatmaps of each cluster as in Figure 6. From the maps, the clerks can then select the most suitable cluster for the new ad, and they can get a reasonable schedule automatically.

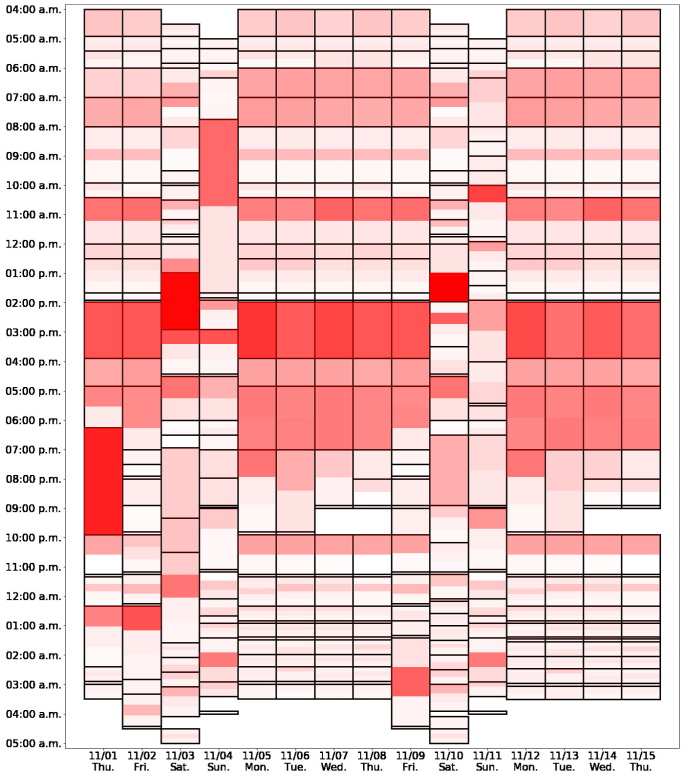

Figure 6: The schedule heatmap of a house cleaning tool TV ad. The depth of the color means the slot preference of the ad. The deeper color is preferred more.

5.2 Scheduling Component Results

We now schedule the ads in the validation data one by one following the ad order described earlier. We compare three scheduling strategies: OR 1, based on the maximization of the total remaining seconds, OR 2, based on minimization of the total extra GRPs, and MaxEnt IO-based scheduling. Note that OR 1 and OR 2 are considered standard TV scheduling strategies based on an operations research approach.

For simplicity, we check the performance on the validation data by picking ads with the highest total GRP requirement. These ads are chosen because having a high total GRP requirement means that the ads have to pay more, so they are considered very valuable to the TV network. Table 2 shows the top 10 members. For the entire validation data, we compare the average distance of the three schedules from the expert ones. The results are shown in Table 2.

As one may expect, OR 1 is realized by satisfying the total GRP requirement with the smallest possible number of slots. To do this, the method prioritizes putting the ads into the popular slots which have high GRPs. As a result, other features related to the effectiveness of broadcasting are neglected, so the distance from the expert schedules is the highest. On the other hand, OR 2 resulted

Table 2: A comparison of the cosine distance ($\times 10^{-7}$) between the expert and generated schedules for the ads with GRP requirements in the top 10 and the overall average in the validation data. Boldface values refer to the best scores.

Products	OR 1	OR 2	MaxEnt IO
Cat food	27.1	3.46	**0.73**
Investments	223	29.0	**8.07**
Hometown tax 1	211	17.1	**6.07**
Internet TV	103	24.4	**14.2**
Hometown tax 2	242	23.6	**7.93**
Delivery pizza	7.65	1.72	**0.99**
Energy drink	169	43.0	**20.3**
Online travel agent 1	194	44.8	**13.5**
Online travel agent 2	158	17.4	**14.5**
Organic vegetable	172	20.0	**0.88**
Overall average	25.6	6.01	**3.66**

in schedules closer to the experts' than OR 1. However, it tends to use the slots which have small GRPs since they are easy to use for acquiring just the required amount of GRPs. Thus, it uses too many slots to meet the requirement, and the scores are relatively higher.

Finally, one sees that the MaxEnt IO-based scheduler generated schedules that have significantly better scores than the standard approaches in all the ads. The proposed data-driven approach is the best in producing schedules that resemble those of the experts, and the results demonstrate our system's potential in improving the TV clerks' scheduling workflow.

6 CONCLUSION

We proposed a data-driven approach to the automation of TV ad scheduling systems. We developed an inverse optimization method which learns expert objectives, which are then used in a mathematical optimization problem we formulated to automatically generate schedules that resemble actual TV clerks' schedules. In addition, we solved an associated cold start problem related to ad requests from new companies or products by applying a suitable clustering method and recovering intentions for the learned clusters. Finally, we demonstrated the effectiveness of our proposed system in automating scheduling and imitating expert decision-making using actual data from a TV network in Japan. Due to its promising results, our proposed system is being prepared for commercial deployment. Future work includes automating cluster assignments using natural language processing techniques and evaluating an online implementation of the proposed system in TV networks.

ACKNOWLEDGMENTS

We would like to thank the TV clerks who provided valuable data and workflow details. We would also like to thank Riki Eto, Yuta Ashida, Dai Kubota, Yuki Nakaguchi and Kei Takemura of NEC Corporation for helpful discussions.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning, ICML 2004*.
[2] Ravindra K. Ahuja and James B. Orlin. 2001. Inverse Optimization. *Operations Research* 49, 5 (2001), 771–783.
[3] Anil Aswani, Zuo-Jun Max Shen, and Auyon Siddiq. 2018. Inverse Optimization with Noisy Data. *Operations Research* 66, 3 (2018), 870–892.
[4] Andreas Bärmann, Sebastian Pokutta, and Oskar Schneider. 2017. Emulating the Expert: Inverse Optimization through Online Learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*. 400–410.
[5] Srinivas Bollapragada, Michael R. Bussieck, and Suman Mallik. 2004. Scheduling Commercial Videotapes in Broadcast Television. *Operations Research* 52, 5 (2004), 679–689.
[6] Srinivas Bollapragada, Hong Cheng, Mary Phillips, Marc Garbiras, Michael Scholes, Tim Gibbs, and Mark Humphreville. 2002. NBC's Optimization Systems Increase Revenues and Productivity. *Interfaces* 32, 1 (Jan. 2002), 47–60.
[7] Srinivas Bollapragada and Marc Garbiras. 2004. Scheduling Commercials on Broadcast Television. *Operations Research* 52, 3 (2004), 337–345.
[8] Timothy C. Y. Chan and Taewoo Lee. 2018. Trade-off preservation in inverse multi-objective convex optimization. *European Journal of Operational Research* 270, 1 (2018), 25–39.
[9] Chaosheng Dong, Yiran Chen, and Bo Zeng. 2018. Generalized Inverse Optimization through Online Learning. In *Annual Conference on Neural Information Processing Systems, NeurIPS 2018*. 86–95.
[10] Chaosheng Dong and Bo Zeng. 2018. Inferring Parameters Through Inverse Multiobjective Optimization. *arXiv preprint arXiv:1808.00935* (2018).
[11] Peyman Mohajerin Esfahani, Soroosh Shafieezadeh-Abadeh, Grani Adiwena Hanasusanto, and Daniel Kuhn. 2018. Data-driven inverse optimization with imperfect information. *Mathematical Programming* 167, 1 (2018), 191–234.
[12] Brian S. Everitt, Sabine Landau, and Morven Leese. 2009. *Cluster Analysis* (4th ed.). Wiley Publishing.
[13] Paul W. Farris, Neil T. Bendle, Phillip E. Pfeifer, and David J. Reibstein. 2010. *Marketing Metrics: The Definitive Guide to Measuring Marketing Performance* (2nd ed.). Wharton School Publishing.
[14] Kimia Ghobadi, Taewoo Lee, Houra Mahmoudzadeh, and Daria Terekhov. 2018. Robust inverse optimization. *Operations Research Letters* 46, 3 (2018), 339–344.
[15] Dentsu Inc. 2018. Advertising Expenditures in Japan. http://www.dentsu.com/knowledgeanddata/ad_expenditures/ (Accessed on 05/15/2019).
[16] Edwin T. Jaynes. 1957. Information Theory and Statistical Mechanics. *Physical Review* 106 (May 1957), 620–630. Issue 4.
[17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. 2001–. SciPy: Open source scientific tools for Python. http://www.scipy.org/ (Accessed on 05/15/2019).
[18] Rick Kurnit. 2018. *Getting the Deal Through - Advertising and Marketing 2018*. Law Business Research Ltd.
[19] Gurobi Optimization LLC. 2019. Gurobi Optimizer Reference Manual. http://www.gurobi.com (Accessed on 05/15/2019).
[20] Andrew Y. Ng and Stuart J. Russell. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000*. 663–670.
[21] Mark J Panaggio, Pak-Wing Fok, Ghan S Bhatt, Simon Burhoe, Michael Capps, Christina J Edholm, Fadoua El Moustaid, Tegan Emerson, Star-Lena Estock, Nathan Gold, Ryan Halabi, Madelyn Houser, Peter R Kramer, Hsuan-Wei Lee, Qingxia Li, Weiqiang Li, Dan Lu, Yuzhou Qian, Louis F Rossi, Deborah Shutt, Vicky Chuqiao Yang, and Yingxiang Zhou. 2016. Prediction and Optimal Scheduling of Advertisements in Linear Television. *arXiv preprint arXiv:1608.07305* (2016).
[22] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[23] Dana G. Popescu and Pascale Crama. 2016. Ad Revenue Optimization in Live Broadcasting. *Management Science* 62, 4 (2016), 1145–1164.
[24] Deepak Ramachandran and Eyal Amir. 2007. Bayesian Inverse Reinforcement Learning. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*. 2586–2591.
[25] Lior Rokach and Oded Maimon. 2005. Clustering Methods. In *The Data Mining and Knowledge Discovery Handbook*. 321–352.
[26] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint* abs/1609.04747 (2016).
[27] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. 2015. Maximum Entropy Deep Inverse Reinforcement Learning. *arXiv preprint arXiv:1507.04888* (2015).
[28] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*. 1433–1438.

A DATA DESCRIPTION

Considering its confidential nature, we present here general and abstract information on the ad scheduling dataset we used in the main paper.

A.1 Slot Data

The TV company plans the schedule of future programs first before it considers the slots in each program. The TV clerks then put the TV ads into the slots based on the following information:

- The parent program's name.
- The slot start and end time.
- The duration in seconds.
- The *slot type*: Slots between two different programs and between program contents are distinguished. The former is called station break (SB) slots and the latter is called participation (PT) slots. Some programs have other types of slots, like those just before and after the programs. The former is called cowcatcher (CC) slots and the latter is called hitchhike (HH) slots.
- The *time rank*: The slots are divided into 4 ranks according to their GRPs: we have A, super B, B and C in descending order. In general, the slots between 7 p.m. to 10 p.m. have the highest GRPs, so their ranks are A. On the other hand, the midnight slots have the lowest GRPs, and therefore their ranks are C. The time rank is a function of the time and day of the week.

A.2 TV Ad Data

This data contains requirements for broadcasting TV ads from the advertisers. The TV clerks create the schedules to satisfy as many requirements as possible. The data consists of the following:

- The company name and contract name.
- The contract start and end date.
- (Optional) The contract start and end time. The advertisers may indicate the specific time to start and end the broadcasting.
- The minimum total GRP requirement.
- (Optional) The free format constraints. Advertisers can give more detailed requirements in the free format section. Due to the difficulty in preprocessing this information, this type of constraint is not used in this paper. If available, this dataset contains the following:
  - The target audience.
  - The GRP ratio throughout the contract period. For example, 30% of the total GRPs should be assigned in the first half, and the remaining 70% should be acquired in the latter half.
  - The preferred time and day of week.
  - The preferred and undesirable program genres or specific program names, actors and actresses. For example, insecticide ads should not be placed in cooking programs.
  - The preferred and undesirable slot types.
- The actual GRP point. Each TV ad indicates the point when to use the actual GRP to estimate the total acquired GRPs.
- The duration in seconds. Almost all TV ads run for only 15 or 30 seconds, but some TV ads have several variations.
- (Optional) The prohibited period. Advertisers can indicate the specific period when the broadcasting of the ads is prohibited in the contract period.
- (Optional) The time constraints. Advertisers can indicate the specific time when the ads can be broadcasted strictly.

A.3 GRP Data

The GRP is a function of GRP points (when to use the actual GRPs) and the slots. We used large table data which contains the values of all combinations of the points and the slots.

A.4 Expert Schedule Data

An expert schedule is just a matrix where each element means the number of putting the TV ad into the slot. In addition, we have timestamp data related to when the ads were put into the slots for each ad and slot pair. Therefore, the expert schedules can be reproduced at any time.

B EXPERIMENTAL DETAILS

Here we describe the experimental details which were omitted in the main paper to encourage reproducibility.

B.1 Train / Test Splits

The target ads we used have contract periods belonging to the duration 2018/07-2018/12. The total number of TV ads is 3,260. An order to the ads is given based on the timestamps they were submitted to the advertisers. As time series data, we used the initial 70% for training, and the remaining 30% for validation. The numbers of ads for training and testing are then 2,282 and 978, respectively.

B.2 TV Ad Clustering

To group the TV ads, we extracted the advertiser features ϕ_{adv} from each expert schedule in the training data and performed clustering.

We applied the agglomerative hierarchical clustering algorithm, which is implemented in SciPy [17] version 1.0.0. The linkage method chosen is complete, the metric is cosine distance and the other options are kept default.

To get the ad clusters, we set the threshold value as 0.3, resulting in 64 clusters.

B.3 MaxEnt IO

To get the objective functions for each cluster, we applied the MaxEnt IO algorithm for the clusters individually. Here we describe some of options or parameters in Algorithm 2.

- Parameter initialization. We set the initial parameter to be 1 in only the first TV feature (the total remaining seconds of all slots after scheduling) while the others are set to 0. Therefore, the first objective for creating schedules in the learning component is the maximization of remaining seconds, which is also described as OR 1 in Section 5.2.
- Number of parameter updates. We set this to 200. Some learning curves, however, start to oscillate near this number like in Figure 7.
- Optimizer or forward solver. When solving the scheduling problem given an estimated parameter, we used the Gurobi

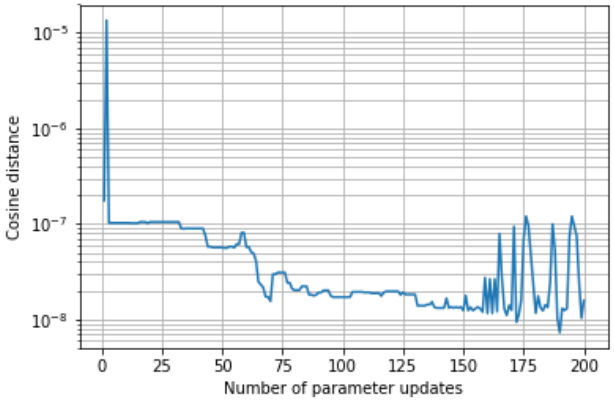

Figure 7: The learning curve of a consumer finance ad.

Optimizer [19] version 6.0. Gurobi was also used in the validation scheduling phase.

- Step size. We fixed it as 1 throughout the learning phase as it gives the most reasonable behavior among trials. For example, we tried another step size, $\alpha = 1/\sqrt{t}$, and a comparison is shown in Figure 8.

B.4 Cluster Assignment

In the experiment, we trained a classifier for assigning new ads in the validation data to learned clusters by treating the clusters obtained from the TV ad clustering as the supervised training data. For training (given 2,282 TV ads), we used the K-Nearest Neighbors algorithm with the cosine distance in scikit-learn [22], version 0.19.1.

To choose the most suitable K, we applied the stratified K-fold cross validation, which is also implemented in scikit-learn [22]. The number of splits is chosen to be 10.

As a result, $K = 1$ gives the highest accuracy of about 72%.

B.5 Scheduling

After cluster assignment, the algorithm executes ad scheduling by using parameters learned by MaxEnt IO. We used the parameters which got the smallest difference between the expert features and the ones extracted from the generated schedules during learning. For example, for cluster B in Figure 4 and Table 1, the 156th parameter was chosen to create the schedules (see Figure 5 for details).

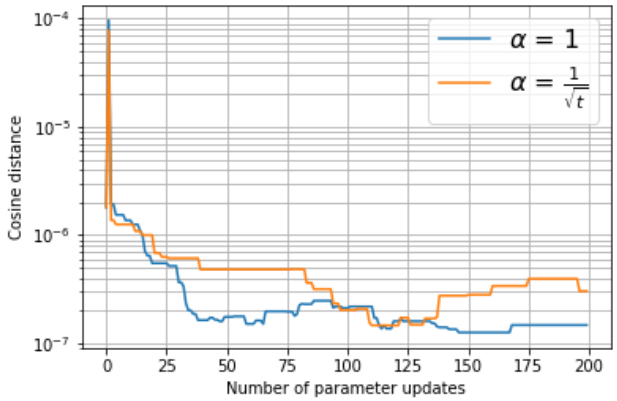

Figure 8: The learning curve of a house cleaning tool ad. The blue curve is the one corresponding to $\alpha = 1$ (this is the same as Figure 5). The orange curve is for the case where $\alpha = 1/\sqrt{t}$.