(I Can Get) Satisfaction: Preference-based Scheduling for Concert-Goers at Multi-Venue Music Festivals

[†]Eldan Cohen, [‡]Guoyu Huang, and [†]J. Christopher Beck

Department of Mechanical & Industrial Engineering
University of Toronto, Toronto, Canada
[†]{ecohen, jcb}@mie.utoronto.ca, [‡]guoyu.huang@mail.utoronto.ca

Abstract. With more than 30 million attendees each year in the U.S. alone, music festivals are a fast-growing source of entertainment, visited by both fans and industry professionals. Popular music festivals are large-scale events, often spread across multiple venues and lasting several days. The largest festivals exceed 600 shows per day across dozens of venues. With many artists performing at overlapping times in distant locations, preparing a personal schedule for a festival-goer is a challenging task. In this work, we present an automated system for building a personal schedule that maximizes the utility of the shows attended based on the user preferences, while taking into account travel times and required breaks. Our system leverages data mining and machine learning techniques together with combinatorial optimization to provide optimal personal schedules in real time, over a web interface. We evaluate MaxSAT and Constraint Programming formulations on a large set of real festival timetables, demonstrating that MaxSAT can provide optimal solutions in about 10 seconds on average, making it a suitable technology for such an online application.

1 Introduction

In recent years, music festival have been growing in popularity, generating significant revenue [14–16]. In the U.S. alone, over 30 million people attend music festivals each year, with more than 10 million attending more than one festival each year [17].

Modern music festivals are large-scale events consisting of a set of musical shows, scheduled over the course of a few days at several different venues. Preparing a personal schedule for a music festival is a challenging task due to the existence of time conflicts between shows and travel times between venues. Festival-goers often spend a significant amount of time deciding which shows to attend, while trying to account for their musical preferences, travel times, and breaks for eating and resting. This problem is often discussed in the entertainment media:

> "The majority of the major conflicts come late in each day—will you dance to HAIM or Flume on Sunday? Will you opt for the upbeat melodies of St. Lucia or Grimes on Saturday?" [12]
>
> "Just when Coachella is upon us and you couldn't be more excited, a cloud enters – the set times are out, and there are heartbreaking conflicts. Difficult decisions must be made. Do you pass over an artist you love because an artist you love even more is playing all the way across the fest?" [13]

In this work, we address the problem of building an optimal schedule based on user preferences. We present a system that uses combinatorial optimization and machine learning techniques to learn the user musical preferences and generate a schedule that maximizes the user utility, while taking into account travel times and required breaks. Our system is implemented over a web interface and is able to generate optimal schedules in less than 10 seconds.

2 Problem Definition

The problem we address consists of two subproblems that need to be solved sequentially. The preference learning subproblem consists of predicting the user's musical preferences based on a small sample of preferences provided by the user. The scheduling subproblem then consists of finding an optimal personal schedule based on the user's preferences.

We first define the problem parameters and then the two subproblems.

2.1 Parameters

Shows. We consider a set of n festival shows $S := \{s_1, s_2, ..., s_n\}$, each associated with one of the performing artists (or bands) in the festival and taking place in one of the festival venues $V := \{v_1, v_2, ..., v_{|V|}\}$. Each show $s_i \in S$ has a fixed start time, t_i^s, and a fixed end time, t_i^e, such that the show length is $t_i^l = t_i^e - t_i^s$.

Travel Times. We consider an $n \times n$ travel time matrix TT, such that TT_{ij} is the travel time between the venue of show s_i and the venue of show s_j. We do not restrict TT to be symmetric, however, we assume it satisfies the triangle inequality.

Show Preferences. To represent the user's musical preferences, we consider the tuple $\langle f_p, M, N \rangle$. $f_p : S \to \mathbb{Z}^+ \cup \{\perp\}$ is a mapping from a show to either an integer score or the special value $\perp$ indicating that the user did not provide a score for the show. $M := \{m_1, m_2, ..., m_{|M|}\}$ is a set of show groups, $m_i \subseteq S$, such that the user must attend at least one show in each group. These groups can be used to model a simple list of shows the user has to attend (i.e., if each group is a singleton), as well as more sophisticated musical preferences such as seeking a diversity of musical styles by grouping shows based on style. Finally, $N \subseteq S$ is a set of shows the user is not interested in attending.

Break Preferences. We consider a set of l required breaks $B := \{b_1, b_2, ..., b_l\}$, such that for each $b_k \in B$, w_k^s and w_k^e represent the start and end of a time window in which the break should be scheduled and w_k^t represents the required break length. We assume that breaks are ordered temporally by their index, the time windows are non-overlapping, and at most one break can be scheduled between each pair of consecutive scheduled shows. The breaks are not allocated to a specific venue: the user can choose to enjoy a break at any location (e.g., at one of the venues or on the way to the next venue). The purpose of scheduling the breaks is to guarantee that sufficient free time is reserved for each b_k during the requested time window.

2.2 Preference Learning Subproblem

Given a function $f_p : S \to \mathbb{Z}^+ \cup \{\perp\}$ that maps shows to scores, our preference learning problem consists of replacing each $\perp$ value by an integer value to produce a full mapping f_p^* that is consistent with the user's preferences. Formally, our problem consists of finding a function $g : S \to \mathbb{Z}^+$ that minimizes the mean squared error [10], a common measure of fit, over the set of shows for which a score was provided $Q = \{s_i \mid f_p(s_i) \neq \perp\}$:

$$min \ \frac{1}{|Q|} \sum_{s_i \in Q} (g(s_i) - f_p(s_i))^2$$

The function g will then be used to predict the missing scores:

$$f_p^*(s) = \begin{cases} f_p(s), & \text{if } f_p(s) \neq \perp \\ g(s), & \textit{if } f_p(s) = \perp \end{cases}$$

2.3 Scheduling Subproblem

Our scheduling subproblem consists of finding an assignment of values for a set of boolean variables $\{x_i \mid i \in [1..n]\}$, representing whether or not the festival-goer attends show s_i, and a set of integer values $\{y_j \mid j \in [1..l]\}$, specifying the start time of break b_j. The assignment has to satisfy the user preference w.r.t. M, N, and B (i.e., groups, shows not attended, and break time-windows). Our objective is to maximize the sum of the user-specified scores for the attended shows:

$$max \sum_{s_i \in S} x_i \cdot f_p^*(s_i)$$

3 System Architecture

The proposed system architecture is illustrated in Figure 1. Our system implements a web interface, accessible using any web-enabled device.

Given an input of user preferences (f_p, M, N, B), provided over a web interface, we start by populating the missing scores in f_p using our preference learning

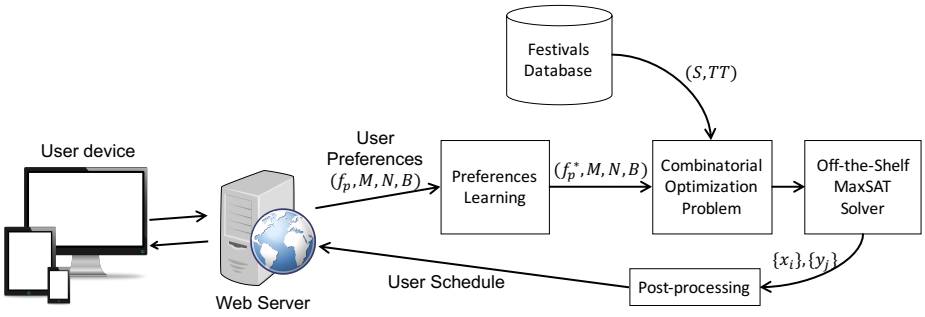

Fig. 1. The system architecture

algorithm. Then, we formulate the scheduling problem as a MaxSAT problem and solve it using an off-the-shelf MaxSAT solver. The details of the shows, S, and the travel time matrix, TT, for all festivals are stored in a database on the server. The results are processed and a schedule is produced and displayed over the web interface.

In Section 4 we present our MaxSAT model for the scheduling problem, followed in Section 5 with an alternative constraint programming (CP) model for the same problem. In Section 6 we describe our preference learning algorithm. Section 7 describes an empirical evaluation of the system, in which we compare the MaxSAT and the CP model, and compare our preference learning method to a simple baseline.

4 MaxSAT Model

In this section, we present our MaxSAT formulation of the problem. We first describe a boolean formulation and then provide a *weighted partial MaxSAT* encoding of the problem.

4.1 A Boolean Formulation

Consider the scheduling subproblem defined in Section 2.3. The variables x_i that represent show attendance are boolean while the variables y_k that represent the start time of each break are integer.

A time-indexed formulation of the problem would consist of replacing each y_k variable with a set of $y_{k,p}$ boolean variables such that $y_{k,p}{=}1 \iff y_k{=}p$. Assuming a time horizon of H time units, we need $H \times |B|$ variables, to represent the breaks. This size may not be unreasonable but here we develop an equivalent model that does not scale with the horizon length.

For any feasible solution for our problem, we can show that there exists a feasible solution with the same objective value, in which if there exists a break b_k between the shows s_i and s_j, it is scheduled in one of the following positions:

1. Immediately after s_i.
2. At the beginning of b_k's time window.

We prove this observation starting with the introduction of required notation and definitions.

We start with z, an assignment of values to the x_i and y_k variables of our problem, and $b_k \in B$, an arbitrary break scheduled at start time y_k. We use $\texttt{before}_z(k)$ to denote the end time of the latest scheduled show before break b_k and $\texttt{after}_z(k)$ to denote the start time of the earliest scheduled show after break b_k, according to the assignment z. We also use $\texttt{tt}_z(k)$ to denote the travel time between the location of the user at time $\texttt{before}_z(k)$ and the location the user needs to be at time $\texttt{after}_z(k)$.

Definition 1 (Feasible Assignment) *Let z be an assignment of values to variables x_i and y_k. We consider z to be a feasible solution if*

1. *All attended shows are non-overlapping.*
2. *The scheduled breaks and the attended shows do not overlap.*
3. *All scheduled breaks are within their time windows, i.e., $y_k \geq w_k^s$ and $y_k + w_k^t \leq w_k^e$.*
4. *For every break b_k that is scheduled between the attended shows s_i and s_j, there is enough time to travel between the show venues and have the break: $\texttt{after}_z(k) - \texttt{before}_z(k) \geq tt_z(k) + w_k^t$.*

Definition 2 (Earliest-break Assignment) *Let z be a feasible assignment of values to variables x_i and y_k. We consider z^*, to be the earliest-break assignment of z if:*

$$x_i^* = x_i \ \forall s_i \in S$$

$$y_k^* = \begin{cases} \texttt{before}_z(k), & \text{if } \texttt{before}_z(k) \geq w_k^s \\ w_k^s, & \textit{if } \texttt{before}_z(k) < w_k^s \end{cases}$$

Figure 2 demonstrates the two possible locations of *earliest-break* assignments. Note that by definition of z^*, $y_k^* \leq y_k$ (otherwise y_k is not feasible).

Lemma 1. *There exists a feasible assignment z if and only if there exists a feasible earliest-break assignment z^*.*

Proof. **Direction $\Longrightarrow$:** We show that if z is feasible, we can construct a z^* that satisfies all the requirements of a feasible solution.

1. Since z is feasible, the attended shows $\{s_i | x_i = true\}$ are not overlapping (Definition 1). Since z^* is an earliest-break assignment of z, $x_i^* = x_i \ \forall s_i \in S$ (Definition 2). Therefore, the attended shows $\{s_i | x_i^* = true\}$ do not overlap (z^* satisfies requirement 1).
2. Since z^* is an earliest-break assignment of z, $\texttt{before}_z(k) \leq y_k^* \leq y_k$ (Definition 2). Consequently, $y_k^* + w_k^t \leq \texttt{after}_z(k)$. Since the shows and breaks do not overlap in z, they do not overlap in z^* (z^* satisfies requirement 2).

3. Since z^* is an earliest-break assignment of z, $w_k^s \leq y_k^* \leq y_k$ (Definition 2). Consequently, $y_k^* + w_k^t \leq y_k + w_k^t$. Therefore, all breaks in z^* are within their time windows (z^* satisfies requirement 3).
4. $\texttt{before}_z(k)$, $\texttt{after}_z(k)$, $tt_z(k)$, and w_k^t remain in z^* as they were in z. Therefore, z^* satisfies requirement 4.

Direction $\Longleftarrow$: If exists an earliest-break assignment z^* that is feasible, then $z = z^*$ is a feasible assignment. □

Note that because the cost function depends only on the x_i variables and $x_i^* = x_i$ for all i, z^* has the same objective value.

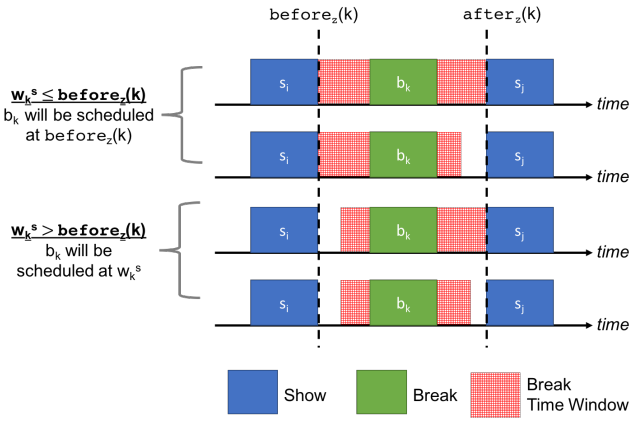

Fig. 2. The two possible locations of *earliest-break* assignments.

We can, therefore, translate the integer start time of a break to a set of boolean variables representing the two possible locations for each break: $q_{k,i}$ that represents whether break b_k is scheduled immediately after show s_i and r_k that represents whether break b_k is scheduled at the beginning of its time window.

4.2 Weighted Partial MaxSAT Encoding

We present a weighted partial MaxSAT model in Figure 3, using *soft clauses* to model the objective function by attaching a weight for each show that corresponds to the user preferences. In our model, (c, w) denotes a clause c with a weight w, while (c, ∞) denotes that c is a *hard-clause*.

We use $R(k)$ to denote the set of all pairs of shows $s_i, s_j \in S$, such that the break b_k cannot be scheduled between the s_i and s_j, due to the necessary transition time. Formally, $R(k)=\{(i,j) \mid t_i^s \leq t_j^s \wedge t_j^s - t_i^e \leq TT_{ij} + w_k^t\}$.

We use the following boolean decision variables in our MaxSAT formulation:

x_i := show $s_i \in S$ is attended.

$q_{k,i}$:= break b_k is scheduled immediately after show s_i.

r_k := break b_k is scheduled at the beginning of its time window.

$$
\begin{array}{lll}
(\neg x_i \vee \neg x_j, \infty) & \forall s_i, s_j \in S : t_i^s \leq t_j^s \wedge t_i^e + TT_{ij} \geq t_j^s & (1) \\
(\bigvee_{x_i \in m_i} x_i, \infty) & \forall m_i \in M & (2) \\
(\neg x_i, \infty) & \forall s_i \in N & (3) \\
(\neg q_{k,i}, \infty) & \forall s_i \in S, b_k \in B : t_i^e + w_k^t \geq w_k^e & (4) \\
(\neg q_{k,i}, \infty) & \forall s_i \in S, k \in B : t_i^e \leq w_k^s & (5) \\
(\neg x_i \vee \neg r_k, \infty) & \forall s_i \in S, b_k \in B : t_i^s \leq w_k^s \leq t_i^e & (6) \\
(\neg x_i \vee \neg r_k, \infty) & \forall s_i \in S, b_k \in B : t_i^s \leq w_k^s + w_k^t \leq t_i^e & (7) \\
(\neg x_i \vee \neg r_k, \infty) & \forall s_i \in S, b_k \in B : w_k^s \leq t_i^s \wedge w_k^s + w_k^t \geq t_i^e & (8) \\
(x_i \vee \neg q_{k,i}, \infty) & \forall i \in S, b_k \in B & (9) \\
(\neg x_i \vee \neg x_j \vee \neg q_{k,i}, \infty) & \forall b_k \in B, (i,j) \in R(k) & (10) \\
(\neg x_i \vee \neg x_j \vee \neg r_k, \infty) & \forall b_k \in B, (i,j) \in R(k) : t_i^e \leq w_k^s \leq t_j^s & (11) \\
(r_k \vee q_{k,1} \vee q_{k,2} \vee ... \vee q_{k,n}, \infty) & \forall b_k \in B & (12) \\
(x_i, f_p(s_i)) & \forall s_i \in S & (13)
\end{array}
$$

Fig. 3. MaxSAT model

Constraint (1) makes sure that for every pair of shows that cannot both be attended (either because they are overlapping or because of insufficient travel time), at most one will be attended. Constraint (2) ensures the shows that the user must attend are part of the schedule and constraint (3) ensures the shows the user is not interested in attending are not part of the schedule.

Constraints (4) and (5) ensure that a break is not scheduled *immediately after* a show, if it means that the break start time or end time is not inside the break's time window.

Constraints (6), (7), (8) ensure that a break is not scheduled at the beginning of its time window if it overlaps an attended show. Constraints (9) ensures that a break is not scheduled *immediately after* a show that is not attended.

Constraint (10) and (11) ensure that a break is not scheduled in any of the two possible locations between two attended shows, if there is not sufficient time.

Constraint (12) ensures that all breaks are scheduled at least once (if a break can be scheduled more than once while maintaining optimality we arbitrarily select one). Constraint (13) is the only soft constraint and is used for optimization. Each clause corresponds to a show with weight equal to the show's score.

5 Constraint Programming Model

We present a CP model in Figure 4. For this model, we utilize optional interval variables, which are decision variables whose possible values are a convex interval: $\{\perp\} \cup \{[s, e) \mid s, e \in \mathbb{Z}, s \leq e\}$. s and e are the start and end times of the interval and $\perp$ is a special value indicating the interval is not present in the solution [9]. The variable $\texttt{pres}(var)$ is 1 if interval variable var is present in the solution, and 0 otherwise. Model constraints are only enforced on interval variables that are present in the solution. $\texttt{start}(var)$ and $\texttt{end}(var)$ return the integer start time and end time of the interval variable var, respectively.

We use the following decision variables in our CP formulation:

$x_i :=$ ($interval$) present if the user attends show s_i and absent otherwise,
$y_k :=$ ($interval$) always present interval representing b_k such that $\texttt{start}(y_k)$ and $\texttt{end}(y_k)$ represent the start and end time of break b_k, respectively.

$$\max \quad \sum_{s_i \in S} f_p(s_i) \cdot \texttt{pres}(x_i) \tag{1}$$

$$\text{s.t.} \quad \texttt{start}(x_i) = t_i^s \qquad \forall s_i \in S \tag{2}$$

$$\texttt{end}(x_i) = t_i^e \qquad \forall s_i \in S \tag{3}$$

$$\sum_{s_i \in m_j} \texttt{pres}(x_i) \geq 1 \qquad \forall m_j \in M \tag{4}$$

$$\texttt{pres}(x_i) = 0 \qquad \forall s_i \in N \tag{5}$$

$$\texttt{start}(y_k) \geq w_k^s \qquad \forall b_k \in B \tag{6}$$

$$\texttt{end}(y_k) \leq w_k^e \qquad \forall b_k \in B \tag{7}$$

$$\texttt{end}(y_k) - \texttt{start}(y_k) = w_k^t \qquad \forall b_k \in B \tag{8}$$

$$\texttt{noOverlap}([x_1, x_2, ..., x_n], TT) \tag{9}$$

$$\texttt{pres}(x_i) \wedge \texttt{pres}(x_j) \Rightarrow \quad \forall b_k \in B; (i,j) \in R(k) \tag{10}$$
$$(\texttt{end}(y_k) \leq \texttt{start}(x_i)) \vee (\texttt{start}(y_k) \geq \texttt{end}(x_j))$$

$$\texttt{start}(y_k) = w_k^s \vee \texttt{start}(y_k) \in \{t_i^s \mid s_i \in S, w_k^s \leq t_i^s \leq w_k^e\} \quad \forall b_k \in B \tag{11}$$

Fig. 4. Constraint programming model

Objective (1) maximizes the user's utility, by summing the utility values of the attended shows. Constraints (2) and (3) set the shows' interval variables, of fixed duration and at fixed times, based on the festival schedule. Constraint (4) ensures that for every group in M, at least one show is attended and constraint (5) ensures the shows the user is not interested in attending are not attended.

Constraints (6) and (7) define the time window for each break based on the user's request by setting a lower bound on the start time and an upper bound on the end time of each break. Constraint (8) defines the length of the break based on the requested break length.

Constraint (9) is responsible for enforcing the feasibility of attending the chosen shows. We use the `noOverlap` global constraint which performs efficient domain filtering on the interval variables with consideration for transition times [9]. Here, the `noOverlap` global constraint ensures that the attended shows do not overlap in time, with consideration for the transition time between the different venues. For example, two consecutive shows can be attended if they are in the same venue. However, if the transition time between the venues is larger than the time difference between the end time of the first show and the start time of the following show, it is impossible to attend both shows.

Constraint (10) ensures the consistency of the break schedule with the attended shows. It verifies that a break is not scheduled between a pair of attended shows that does not have sufficient time difference to allow traveling and taking the break. For every triplet of two attended shows and a break such that the break cannot be scheduled between the two shows due to time (i.e., the time difference between the two shows is smaller than the sum of the transition time and the break time), we make sure the break is either scheduled before the earlier show or after the later show. Note that simple `noOverlap` constraints are not sufficient as we need to ensure the break will be scheduled either before the earlier show or after the later one and there is no required transition time between the end (resp., start) of a show and the start (resp. end) of a break.

Constraint (11) reduces the domains of each break variable, based on Lemma 1. While it does not enforce an *earliest-break assignment* (not required), it does reduce the domains of the break interval variables by allowing each break to be scheduled only at the beginning of the break's time window, or immediately after shows that end in the break's time window.

6 Learning User Preferences

The user cannot be expected to know all the performing artists in a festival or to spend time assigning a score to every artist. Therefore, we employ a learning-based approach to populate the missing scores based on the user's assignment of scores to a subset of the shows. We leverage the availability of large on-line music datasets to mine the features to predict the user's score.

As every show is associated with a performing artist, we assume the score assigned to each show reflects the user preference for the musical style of the performing artist. Therefore, our approach is to use the tags assigned to each artist on Last.fm,[1] a popular music website, as a feature set for a regression model that predicts the user score. The tags typically describe the artist musical style and origin (e.g., *pop*, *indie rock*, *punk*, *australian*, *spanish*, etc). Tag-based features have been shown to be successful in recommending music [6, 11]. In this work, we use linear regression model, due to its simplicity and its success in tag-based recommendation systems [22].

Given a training set of K inputs of $(\overrightarrow{o_i}, p_i)$ for $1 \leq i \leq K$ where $\overrightarrow{o_i}$ if a vector of F features and $p_i \in \mathbb{R}$ is a real value, the regression problem finds a set of

[1] http://www.last.fm

weights, β_i, that expresses the value as a linear combination of the features and an error ε_i: $p_i = \beta_0 + \beta^1 o_i^1 + \beta^2 o_i^2 + ... + \beta^K o_i^K + \varepsilon_i$, such that the mean squared error over the training set is minimized [21]:

$$min\ \frac{1}{K}\sum_{i=1}^{K}\varepsilon_i^2$$

We collect the tags associated with all performing artists in the festival using Last.fm API.[2] For each artist, we construct a binary feature vector describing whether a tag applies to an artist. We train our model based on the subset of artists for which a score has been provided by the user. We then predict scores for the rest of performing artists, based on their associated tags. The predicted scores are rounded to integers as per the problem definition.

Due to the properties of this problem, notably a large number of features compared to a small training set, we choose to use Elastic Nets [26]. Elastic Nets employ a convex combination of L_1 and L_2 regularization using a parameter α, such that $\alpha = 1$ corresponds to ridge regression [8] and $\alpha = 0$ corresponds to lasso regression [24]. We use 5-fold cross-validation on the training set to choose the α value from a set of 10 values in $[0, 1]$. To reduce training time, we start by performing a univariate feature selection based on the F-test for regression [4], i.e., for each feature we calculate the F-test value for the hypothesis that the regression coefficient is zero, and select the top 75 features.

For comparison, we also consider Support Vector Regression (SVR). In our empirical evaluation we compare the linear regression based on Elastic Nets, to a linear SVR model. Linear SVR are much faster to train and to generate predictions than nonlinear SVR, and often give competitive results [7], making them an interesting candidate for our application. The regularization parameter and the loss function are chosen using a 5-fold cross validation on the training set.

7 Empirical Evaluation

In this section we present an empirical evaluation of our system. As our work consists of two parts – preference learning and scheduling, we evaluate each part separately.

A thorough evaluation of our application would require performing an experiment with real users to evaluate their satisfaction of the system. Unfortunately, such experiment is outside the scope of this research at this time. Instead, we leverage the existence of on-line datasets to empirically evaluate our system using real data.

First, we perform an experiment that evaluates the success of our preference learning method in predicting the musical taste of real users. Then, we perform an empirical evaluation of the scheduling system based on the timetables of real music festivals.

[2] http://www.last.fm/api

7.1 Preference Learning Evaluation

In this section we present an experiment designed to evaluate the success of our preference learning method in predicting the musical taste of a user. The preference learning method we develop in Section 6 takes in a training set of tuples $(user, artist, score)$ and predicts the score for $(user, artist)$ tuples using tags from Last.fm. To evaluate this method without real users, we need a way to generate potential user scores on a subset of artists as an input to our learning algorithm and then a way to find a set of "ground truth" scores for the rest of the artists to compare against our predicted scores.

We could not find a dataset of this form that we can directly apply our method on. Instead, we take an existing dataset describing the listening habits of users and manipulate it into the required form of $(user, artist, score)$. We then split it to a training set used to train our learning algorithm and a test set on which we can compare our predicted score.

Data Preparation. The chosen dataset r is a collection of tuples $(user, artist, \#plays)$ describing the listening habits of 360,000 users on Last.fm, collected by Celma [3]. Each tuple describes the number of times each user listened to songs of each artist. For this experiment, we sample a set of 1000 users, U, each with at least 50 tuples. On average, each user in our sample has tuples for 57.85 artists.

In order to transform this dataset to the required form, we need to substitute the $\#plays$ column with a score column. To do so, we sort each user's records based on $\#plays$, and provide a score between 1 to 8, based on the corresponding quantile of $\#plays$. The artists with the lowest $\#plays$ will have a score of 1, and the artists with the highest $\#plays$ will have a score of 8. For convenience, we refer to the transformed dataset as a set of 1000 *user-specific datasets*. Given a user $u \in U$, we use r_u to denote the dataset that consists of $(artist, score)$ tuples that correspond to the user's tuples in r, $r_i = \Pi_{\langle artist, score \rangle}(\sigma_{user=u}(r)) \ \ \forall u \in U$.

Experiment Setup. To evaluate the preference learning method, we split each user's transformed dataset into a training set (60%) and a test set (40%). Given approximately 58 tuples per user, we get an average training set of approximately 35 records. We consider this to be a reasonable number of input scores to expect from a music festival attendee.

For each user, we train the model described in Section 6 based on the training set and then measure the *Minimum Squared Error* (MSE) and *Minimum Absolute Error* (MAE) in predicting the score of the records in the test set. We then calculate the median value of MSE and MAE across all users.

We compare our preference learning method, based on Elastic Net, with a linear SVR model and a baseline that consists of substituting all the missing scores with the mean score.

Experiment Results. Table 1 shows the median MSE, the median MAE, and the mean runtime (train + test) across all r_u, for the Elastic Net, the linear SVR,

and the mean baseline. It is clear that the learning based methods significantly outperform the baseline. The Elastic Net method yields more accurate results, however it requires longer runtime. Note that we expect the runtime in our system to be longer due to a larger set of shows for which we must predict scores (all shows in the festival).

Table 1. Median MSE, median MAE and average time across a dataset of 1000 users

Algorithm	Median MSE	Median MAE	Mean Runtime (sec)
Elastic Net	3.45	1.56	0.94
Linear SVR	3.79	1.64	0.35
Mean score baseline	6.24	2.25	0.24

7.2 Schedule Evaluation

In this section we present an empirical evaluation of the scheduling system. All experiments were run on a dual-core i5 (2.7GHz) machine with 16GB of RAM running Mac OS X Sierra. For our MaxSAT model, we used MaxHS v2.9, that employs a hybrid SAT and MIP approach [5]. For our CP model, we used CP Optimizer from the IBM ILOG CPLEX Optimization Studio version 12.6.3, single-threaded with default search and inference settings. We use a 10-minute run-time limit for each experiment.

Problem Instances. We consider 34 instances based on the real timetables of seven popular music festivals in the recent years as shown in Table 2. The instances have a large range of sizes defined by the number of shows, $|S|$, the number of venues, $|V|$, the number of must-attend groups, $|M|$, the number of shows the user is not interested in attending, $|N|$, and the number of required breaks, $|B|$. The parameters S and V are based on the real festival timetable. The travel time matrix TT is randomly generated in a range based on the estimated travel time between the real festival venues and it satisfies the triangle inequality. The breaks in B are generated in two configurations: either 2 breaks of 30 minutes or one break of 60 minutes. The shows in M and N were arbitrarily chosen, discarding infeasible instances. Each M has a mix of singletons and groups with multiple items. Random scores between 1-10 were assigned to a subset of the artists that ranged between 50% for the smallest festival to 15% for the largest one. The missing scores are predicted using our Elastic Net model. The preference learning runtime for each instance ranges between 2 to 6 seconds per user.

Numerical Results. Table 3 shows the time it takes to find and prove an optimal solution. The table is ordered in increasing size of the festivals. For the smaller instances, both MaxSAT and CP find and prove an optimal solution in a short time (usually less than 1 second). For the medium to large instances (starting from Glastonbury), CP struggles to find and prove optimal solutions,

Timetable Source	$\|S\|$	$\|V\|$	$\|M\|$	$\|N\|$	$\|B\|$	Learning Time (sec)
Pitchfork'16 Saturday	17	3	1	1	2	5.72
Pitchfork'16 Sunday	16	3	2	2	1	3.27
Pitchfork'17 Saturday	14	3	1	1	2	2.64
Pitchfork'17 Sunday	14	3	2	2	1	3.38
Lollapalooza Chile'17 Saturday	34	6	1	1	2	2.92
Lollapalooza Chile'17 Sunday	34	6	3	3	1	2.97
Primavera'16 Thursday	35	6	1	1	1	2.65
Primavera'16 Friday	35	6	3	3	2	2.52
Primavera'17 Thursday	35	6	1	1	2	2.42
Primavera'17 Friday	35	6	3	3	1	2.68
Osheaga'15 Friday	40	6	1	1	2	2.73
Osheaga'15 Saturday	38	6	3	3	1	2.29
Osheaga'15 Sunday	38	6	5	5	2	2.63
Osheaga'16 Friday	38	6	1	1	1	3.25
Osheaga'16 Saturday	38	6	3	3	2	2.36
Osheaga'16 Sunday	37	6	5	5	1	2.54
Glastonbury'15 Friday	90	10	1	1	2	3.21
Glastonbury'15 Saturday	92	10	3	3	1	2.91
Glastonbury'15 Sunday	94	10	5	5	2	2.84
Glastonbury'16 Friday	95	10	1	1	1	2.40
Glastonbury'16 Saturday	90	10	3	3	2	2.65
Glastonbury'16 Sunday	90	10	5	5	1	2.60
Tomorrowland'14 #1 Friday	149	15	1	1	2	2.76
Tomorrowland'14 #1 Saturday	139	15	3	3	1	2.51
Tomorrowland'14 #1 Sunday	118	14	5	5	2	2.73
Tomorrowland'14 #2 Friday	149	15	1	1	1	2.58
Tomorrowland'14 #2 Saturday	139	15	3	3	2	2.96
Tomorrowland'14 #2 Sunday	129	15	5	5	1	2.56
SXSW'15 Thursday	685	100	1	1	1	4.21
SXSW'15 Friday	706	102	3	3	2	4.04
SXSW'15 Saturday	715	99	5	5	1	4.19
SXSW'17 Thursday	644	96	1	1	2	3.60
SXSW'17 Friday	564	94	3	3	1	3.62
SXSW'17 Saturday	566	77	5	5	2	3.66
						Mean learning time: 3.03

Table 2. Description of the problem instances: number of shows $|S|$, number of venues $|V|$, number of must-attend groups $|M|$, number of unattended shows $|N|$, and the time it takes to train the learning model and predict the missing scores.

especially for the less constrained instances (i.e., smaller $|M|$ and $|N|$). For the largest music festival (SXSW), CP times-out on some of the instances (denoted "T/O"), failing to prove optimality (although it does *find* an optimal solution in all cases). MaxSAT, however, seems to be able to scale well, with the hardest instance taking approximately 12 seconds to find an optimal solution and prove its optimality.

Table 3 also shows the time it took to find the optimal solution without proving its optimality. MaxSAT still demonstrates better results in most cases, however in almost all cases, CP manages to find the optimal solution in less than one minute.

		Find+Prove Opt. (sec)		Find Opt. (sec)	
Instance	Objective	MaxSAT	CP	MaxSAT	CP
Pitchfork'16 Saturday	40	0.02	0.02	0.00	0.00
Pitchfork'16 Sunday	30	0.02	0.02	0.00	0.00
Pitchfork'17 Saturday	43	0.02	0.01	0.00	0.00
Pitchfork'17 Sunday	39	0.02	0.02	0.00	0.00
Lollapalooza Chile'17 Saturday	34	0.04	0.39	0.00	0.01
Lollapalooza Chile'17 Sunday	33	0.03	0.04	0.00	0.00
Primavera'16 Thursday	50	0.03	0.06	0.00	0.01
Primavera'16 Friday	60	**0.04**	0.48	**0.01**	0.19
Primavera'17 Thursday	36	0.03	0.05	0.00	0.00
Primavera'17 Friday	38	**0.03**	0.39	0.00	0.06
Osheaga'15 Friday	52	0.12	0.91	0.00	0.01
Osheaga'15 Saturday	38	0.02	0.05	0.00	0.01
Osheaga'15 Sunday	33	0.02	0.04	0.00	0.00
Osheaga'16 Friday	51	0.05	0.39	0.00	0.00
Osheaga'16 Saturday	26	0.04	0.09	0.01	0.02
Osheaga'16 Sunday	38	0.03	0.04	0.00	0.00
Glastonbury'15 Friday	64	**0.59**	91.27	**0.38**	15.28
Glastonbury'15 Saturday	67	**0.13**	3.13	0.03	0.20
Glastonbury'15 Sunday	53	0.06	0.20	0.00	0.07
Glastonbury'16 Friday	85	**0.06**	110.78	**0.01**	1.33
Glastonbury'16 Saturday	70	**0.06**	1.90	**0.01**	0.12
Glastonbury'16 Sunday	61	0.18	1.07	0.01	0.05
Tomorrowland'14 #1 Friday	67	**1.81**	532.46	**1.71**	211.99
Tomorrowland'14 #1 Saturday	49	**0.39**	4.08	0.02	0.12
Tomorrowland'14 #1 Sunday	45	0.06	0.34	**0.00**	0.13
Tomorrowland'14 #2 Friday	58	**11.13**	408.23	**0.02**	0.42
Tomorrowland'14 #2 Saturday	48	**0.09**	2.68	**0.01**	0.99
Tomorrowland'14 #2 Sunday	42	0.37	1.15	0.30	0.08
SXSW'15 Thursday	133	**8.00**	T/O	**4.67**	61.10
SXSW'15 Friday	119	**3.00**	T/O	**1.11**	16.41
SXSW'15 Saturday	116	**9.88**	218.35	6.59	8.72
SXSW'17 Thursday	98	**6.82**	T/O	2.73	25.18
SXSW'17 Friday	101	**2.76**	80.04	**0.19**	5.93
SXSW'17 Saturday	125	**4.90**	T/O	**0.57**	96.04
Mean run-time		**1.50**	113.49	**0.54**	13.07

Table 3. Time to find and prove optimal solution and time to find optimal solution for MaxSAT and CP (results that are at least 10 times better are in bold).

8 Related Work

Personal-level scheduling has received little attention in recent optimization literature. Refanidis and Yorke-Smith [20] presented a CP model for the problem of automating the management of an individual's time, noting the problem's difficulty due to the variety of tasks, constraints, utilities, and preference types involved. Alexiadis and Refanidis [2, 1] presented a post-optimization approach, in which an existing personal schedule is optimized using local search. They developed a bundle of transformation methods to explore the neighborhood of a solution using either hill climbing or simulated annealing and achieved more than 6% improvement on average.

Closely-related problems, such as conference scheduling, have only been addressed from the perspective of building the *event schedule* with the objective of either meeting the presenters' or attendees' preferences [23]. An example for a presenter-based perspective approach can be found in Potthoff and Brams's integer programming formulation for conference scheduling w.r.t. presenters' availability [18]. Examples for attendee-based perspective approaches can be found in Quesnelle and Steffy's work on minimizing attendee conflicts [19], using an integer programming model, and Vangerven et al.'s work on maximizing attendance using a hierarchical optimization approach [25]. We are not aware of any work that directly addresses music festival scheduling nor of any work which takes the event schedule as input and optimizes for the individual attendee.

9 Conclusions and Future Work

We present a preference-based scheduling system for concert-goers at multi-venue music festivals. We utilize data mining and machine learning techniques to learn the user preferences and reduce the required user input. We use MaxSAT to efficiently find and prove an optimal schedule that maximizes the user utility, while taking into consideration the travel times between venues and the user's break preferences. Our system implements a web interface in which the user provides the required inputs and accesses the resulting schedule.

Our empirical evaluation shows that the use of preference learning allows us to provide more accurate results and the use of a MaxSAT model allows us to provide an efficient online service, with most instances taking less than 5 seconds and the hardest instances reaching 15 seconds for learning and optimization.

We believe this system can easily be adapted to other kinds of multi-venue events, such as conferences and large sporting events. For example, in the context of a conference, the preference learning can rely on the keywords of each talk and generate a preference-based personal schedule of talks to attend.

Another potential extension of this work is to explore ways to provide the users with alternative schedules. In this work the preference learning method is aimed at finding a schedule that is consistent with the user preferences. However, as some visitors often use the festival to expand their musical horizons, investigating ways to generate schedules that introduce the users to music they are not familiar with is an interesting direction of research.

References

1. Alexiadis, A., Refanidis, I.: Optimizing individual activity personal plans through local search. AI Communications 29(1), 185–203 (2015)
2. Alexiadis, A., Refanidis, J.: Post-optimizing individual activity plans through local search. In: Proceedings of the 8th Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS'13). pp. 7–15 (2013)
3. Celma, Ò.: Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space. Springer (2010)
4. Chatterjee, S., Hadi, A.S.: Regression analysis by example. John Wiley & Sons (2015)
5. Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in MaxSAT. In: International conference on theory and applications of satisfiability testing. pp. 166–181. Springer (2013)
6. Firan, C.S., Nejdl, W., Paiu, R.: The benefit of using tag-based profiles. In: Web Conference, 2007. LA-WEB 2007. Latin American. pp. 32–41. IEEE (2007)
7. Ho, C.H., Lin, C.J.: Large-scale linear support vector regression. Journal of Machine Learning Research 13(Nov), 3323–3348 (2012)
8. Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics 42(1), 80–86 (2000)
9. Laborie, P.: IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. In: International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems. pp. 148–162. Springer (2009)
10. Lehmann, E.L., Casella, G.: Theory of point estimation. Springer Science & Business Media (2006)
11. Levy, M., Sandler, M.: Music information retrieval using social tags and audio. IEEE Transactions on Multimedia 11(3), 383–395 (2009)
12. Long, Z.: Start planning your weekend with the lollapalooza 2016 schedule (May 2016), https://www.timeout.com/chicago/blog/start-planning-your-weekend-with-the-lollapalooza-2016-schedule-050916, [Online; posted May 9, 2016]
13. Lynch, J.: Coachella 2016: 10 heartbreaking set time conflicts (and how to handle them) (May 2016), http://www.billboard.com/articles/columns/music-festivals/7333891/coachella-2016-set-time-schedule-conflicts, [Online; posted April 14, 2016]
14. McIntyre, H.: America's top five music festivals sold $183 million in tickets in 2014 (March 2015), http://www.forbes.com/sites/hughmcintyre/2015/03/21/americas-top-five-music-festivals-sold-183-million-in-tickets-in-2014, [Online; posted Mar 21, 2015]
15. McIntyre, H.: New york city's music festival market is becoming increasingly crowded (June 2016), http://www.forbes.com/sites/hughmcintyre/2016/06/21/new-york-citys-music-festival-market-is-becoming-increasingly-crowded, [Online; posted Jun 21, 2016]
16. Mintel: Music concerts and festivals market is star performer in the uk leisure industry as sales grow by 45% in 5 years (December 2015), http://www.mintel.com/press-centre/leisure/music-concerts-and-festivals-market-is-star-performer-in-the-uk-leisure-industry-as-sales-grow-by-45-in-5-years, [Online; posted Dec 9, 2015]
17. Nielsen: For music fans, the summer is all a stage (April 2015), http://www.nielsen.com/us/en/insights/news/2015/for-music-fans-the-summer-is-all-a-stage.html, [Online; posted Apr 14, 2015]

18. Potthoff, R.F., Brams, S.J.: Scheduling of panels by integer programming: Results for the 2005 and 2006 New Orleans meetings. Public Choice 131(3-4), 465–468 (2007)
19. Quesnelle, J., Steffy, D.: Scheduling a conference to minimize attendee preference conflicts. In: Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA). pp. 379–392 (2015)
20. Refanidis, I., Yorke-Smith, N.: A constraint-based approach to scheduling an individual's activities. ACM Transactions on Intelligent Systems and Technology (TIST) 1(2), 12 (2010)
21. Sen, A., Srivastava, M.: Regression analysis: theory, methods, and applications. Springer Texts in Statistics, Springer, New York (1990), http://cds.cern.ch/record/1611847
22. Sen, S., Vig, J., Riedl, J.: Tagommenders: connecting users to items through tags. In: Proceedings of the 18th international conference on World wide web. pp. 671–680. ACM (2009)
23. Thompson, G.M.: Improving conferences through session scheduling. The Cornell Hotel and Restaurant Administration Quarterly 43(3), 71–76 (2002)
24. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996)
25. Vangerven, B., Ficker, A., Goossens, D., Passchyn, W., Spieksma, F., Woeginger, G.: Conference scheduling: a personalized approach. In: Burke, E., Di Gaspero, L., Özcan, E., McCollum, B., Schaerf, A. (eds.) Proceedings of the 11th International Conference on the Practice and Theory of Automated Timetabling. pp. 385–401. PATAT (2016)
26. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B 67, 301–320 (2005)