MIS Quarterly

RESEARCH ARTICLE

THE RIGHT MUSIC AT THE RIGHT TIME: ADAPTIVE PERSONALIZED PLAYLISTS BASED ON SEQUENCE MODELING[1]

Elad Liebman
SparkCognition, Inc., 4030 West Braker Lane #500, Austin, TX 78759 U.S.A. {eladlieb@gmail.com}

Maytal Saar-Tsechansky
McCombs School of Business, The University of Texas at Austin, 2110 Speedway, Stop B6500, Austin, TX 78712-1277 U.S.A. {maytal@mail.utexas.edu}

Peter Stone
Department of Computer Science, The University of Texas at Austin, 2317 Speedway, Stop D9500, Austin, TX 78712-1277 U.S.A. {pstone@cs.utexas.edu}

Recent years have seen a growing focus on automated personalized services, with music recommendations a particularly prominent domain for such contributions. However, while most prior work on music recommender systems has focused on preferences for songs and artists, a fundamental aspect of human music perception is that music is experienced in a temporal context and in sequence. Hence, listeners' preferences also may be affected by the sequence in which songs are being played and the corresponding song transitions. Moreover, a listener's sequential preferences may vary across circumstances, such as in response to different emotional or functional needs, so that different song sequences may be more satisfying at different times. It is therefore useful to develop methods that can learn and adapt to individuals' sequential preferences in real time, so as to adapt to a listener's contextual preferences during a listening session. Prior work on personalized playlists either considered batch learning from large historical data sets, attempted to learn preferences for songs or artists irrespective of the sequence in which they are played, or assumed that adaptation occurs over extended periods of time. Hence, this prior work did not aim to adapt to a listener's current song and sequential preferences in real time, during a listening session. This paper develops and evaluates a novel framework for online learning of and adaptation to a listener's current song and sequence preferences exclusively by interacting with the listener, during a listening session. We evaluate the framework using both real playlist datasets and an experiment with human listeners. The results establish that the framework effectively learns and adapts to a listeners' transition preferences during a listening session, and that it yields a significantly better listener experience. Our research also establishes that future advances of online adaptation to listener's temporal preferences is a valuable avenue for research, and suggests that similar benefits may be possible from exploring online learning of temporal preferences for other personalized services.

Keywords: Online preference learning, personalized adaptation, user preferences, music playlist preference, reinforcement learning, data science

[1]Sumit Sarkar was the accepting senior editor for this paper. Paul Hu served as the associate editor.

The appendices for this paper are located in the "Online Supplements" section of *MIS Quarterly*'s website (https://misq.org).

Introduction

Music is one of the most prevalent expressions of human culture, with implications for individuals' decision making and overall well-being. Research on music recommendations has focused almost exclusively on learning preferences for songs or artists. However, as an activity, music is also inherently temporal: sounds occur in a *sequence*, and at a higher level, listening sessions span over a sequence of songs (Davies 1978; Kahnx et al. 1997). Indeed, the pleasure from music is also affected by the particular sequence in which songs are being played (Palmer 2005): listeners indicate clear sequential preferences while interacting with a playlist generator (Kaji et al. 2005), and the song sequence and resulting transitions in a given playlist are at the core of how DJs produce a desired listener experience (Cliff 2000). Thus, in contrast with the view of a playlist as a *collection* of desirable songs played in some order, in this paper we consider a playlist as a particular sequence of songs. We first posit that preferences for such sequences affect the listeners' experience, and that if these preferences can be learned, they can be used to plan personalized playlists and improve the listener's experience.

Because a listener's sequential preferences may vary across circumstances, such as in response to different emotional or functional needs, different sequences may be more satisfying in different contexts: one's preferences at the end of a working day may differ on different days, and a listener may enjoy an eclectic sequence with dramatic transitions between songs or smoother transitions between songs at different contexts. Hence, in this paper we aim to learn and adapt to such preferences online, in a given context.

Prior research on learning sequential preferences either considered settings in which a listener's preferences are learned in a batch fashion from historical data (e.g., Chen et al. 2012; Harir et al. 2012; Maillet et al. 2009; McFee and Lanckriet, 2011; Natarajan et al. 2013; Zheleva et al. 2010), which may capture a listener's preferences across multiple contexts, or considered online (real time) learning of and adaptation to preferences based on the Q-learning framework (Sutton and Barto 1998) that requires a large number of interactions with listeners (e.g., over multiple days) before yielding benefits (e.g., Chi et al. 2010; King and Imbrasait 2015). In this paper, we address a fundamentally different task: learning and adapting online to a listener's song and sequential preferences during a listening session that may only last a few hours, so as to tailor a playlist to the listener's songs and sequence preferences in the current context. As such, the framework we develop aims to augment existing recommendation technologies whose goal is to learn a listener's preferences from historical data and across multiple contexts. Henceforth, we use the term *online* to refer to learning and adaptation to the listener through interactions in real time, during a listening session (not from historical data). We use the term *batch learning* to refer to learning from historical data, acquired prior to the listening session.

Our contributions are as follows. First, we develop a novel framework for a personalized DJ, DJ-MONTE CARLO (henceforth, DJ-MC), that learns and adapts the songs and sequence in which they are played to the listeners' preferences in real time, by interacting with the listener during a listening session. To the best of our knowledge, prior research on playlist learning and generation has either focused exclusively on batch learning from historical data, considered listeners' preferences for individual songs or artists irrespective of the sequence in which songs are being played, or considered online adaptation over long periods of time and thus did not aim to suit a listener's preferences in a given context. Consequently, prior research has not explored the design challenges of real-time learning and adaptation to sequential preferences by interacting with the listener during a listening session, nor whether such an adaptation can yield a better listening experience for the listener. In this work, we aim to bridge these gaps. Second, our real-time learning task requires that learning from limited interactions with the listener be highly efficient. We thus identify key design properties for achieving this goal, and we empirically study their contributions to our framework's performance. Importantly, these design choices are generic, and are directly applicable to domains other than music. Third, we propose two extensions of our framework that account for song costs when producing a playlist. We empirically evaluate the benefit of each approach and highlight their distinct properties.

We evaluate the performance of our framework over simulated data derived from real playlists, and we draw insights on its relative performance as compared to alternatives under a controlled environment. We complement this study with another study involving human listeners, and we find that our framework yields significantly more enjoyable sequences to human listeners. This study confirmed the benefits of our approach while it adapts to human listeners, whose music preferences can be heterogeneous and arbitrarily complex.

The challenge we address here generalizes to other settings, where both learning preferences and adapting to them in real time is desirable. These settings include, for example, producing sequential recommendations for news items, videos (e.g., YouTube videos), or even workout exercises. In all of these cases, the items one enjoys and the sequence that would be more pleasing can vary in different contexts. Broadly, this research also contributes to the growing body of work on contextual recommendations, which aims to extend existing recommendation methods by adapting the recommendations to suit a user's current context (e.g., Adomavicius and Tuzhilin 2005; Natarajan et al. 2013).

Finally, the framework we develop builds on the reinforcement learning paradigm for simultaneously learning from interactions and acting in real time to achieve a goal. Reinforcement learning is a natural progression from batch machine learning from historical data, and a part of a continuing progression toward artificial intelligence techniques that learn and act on their own, through interactions with their environment. Because there has been little research in Information Systems (IS) considering these settings, we provide an introduction to the reinforcement learning problem, and hope that this research will contribute to an IS research stream, addressing business, organizational, or societal challenges that can benefit from systems that independently learn and act in their environment.

Reinforcement Learning Preliminaries: Learning Through Interactions

We consider an interactive setting, illustrated in Figure 1, in which a recommender agent plays a sequence of songs to a listener. After each song, the listener provides feedback on the song, such as a numeric score or a like/dislike signal, reflecting the benefits the listener derives from each song; once the feedback is received, the agent's representation of the listener's preferences are updated to reflect the feedback, and the agent then adapts to the listener's preferences by recommending the next song that suits the listener's preferences at the current time. The agent's overall goal is to produce the most pleasing playlist for the listener. Note that each song played may also have long-term consequences on the listener's enjoyment from the remainder of the playlist.

We begin with a brief introduction to reinforcement learning, followed by a definition of our problem within this domain. The problem, described above, is one of reinforcement learning (RL): learning what actions to take in different situations so as to gain maximum reward over a sequence of actions, by trying out different actions and discovering what rewards they produce (Sutton and Barto 1998). The reward, which is typically a real number scalar, reflects the benefits from the agent's recent action. In our music playlist adaptation problem, the reward is a real number, capturing the listener's feedback (reflecting enjoyment from the playlist). The goal of an RL agent is to maximize the cumulative reward over the listening session. RL is thus defined not by the characteristics of how the learning methods solve the problem, but by the problem itself. A reinforcement learning method is thus any method that learns and acts simultaneously via interactive trial and error with delayed rewards so as to achieve a goal. By contrast, the kind of machine learning studied most often is batch learning from historical data, such as supervised learning. However, supervised learning is inadequate for learning from interactions, as entailed in our setting.

The key elements of the RL problem are as follows. A decision-making agent (e.g., DJ-MC) makes a sequence of decisions (song selections): at each step, it observes the state of the environment, decides and takes an action (plays the next song), receives an immediate reward, representing the immediate benefit from the action, and observes the new state of the environment. Importantly, rather than learn from historical data, the only way the agent can learn which actions are good in different situations is to sequentially try them out and receive a reward. The agent's goal is typically that the cumulative reward over the entire sequence is maximized (e.g., maximize the listener's enjoyment from the entire playlist).

Note that in our interactive setting, the agent's action of selecting the next song to play produces an immediate reward that counts toward the agent's cumulative reward. However, an RL agent has two motivations not to play the song that yields the best immediate reward. First, similar to active learning methods (e.g., Saar-Tsechansky et al. 2009; Saar-Tsechansky and Provost 2007), it ought to acquire informative experiences so as to quickly improve its ability to correctly predict the outcome of its actions. In addition, each song played may also have long-term consequences: it may affect how the listener enjoys the next song as well as future songs in the playlist. Hence, it may be beneficial not to play the most enjoyable song next, so as to improve learning and produce a more enjoyable playlist overall.

Finally, to effectively infer the consequences of its actions, such as the future reward from an action, a reinforcement agent must have useful information to make those inferences. Ideally, it is desirable that the agent can represent the history of its interactions in a compact manner, which also retains all relevant information for inferring future states and rewards. As we will see below, in our setting it is possible to retain a representation of the playlist history to yield informative representation of the agent's interaction history. Note that having such complete information merely ensures that the agent has the best possible information from which to select actions, yet inferring the future consequences of the agent's actions for any given situation can still be difficult (Sutton and Barton 1998).

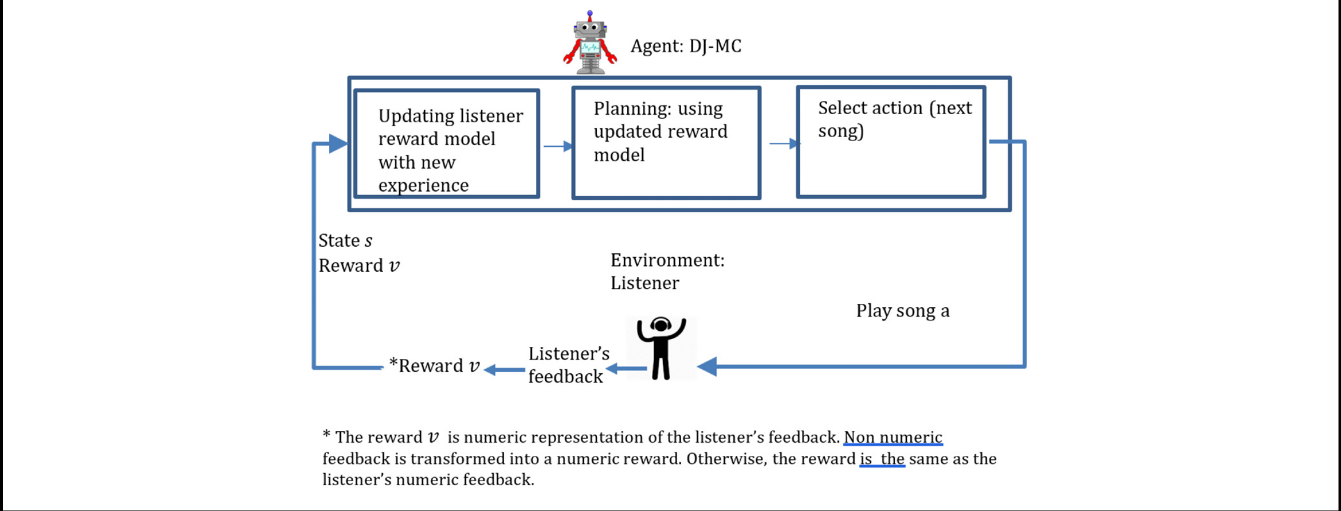

Agent: DJ-MC
Updating listener reward model with new experience
Planning: using updated reward model
Select action (next song)
State s
Reward v
Environment: Listener
Play song a
*Reward v
Listener's feedback
* The reward v is numeric representation of the listener's feedback. Non numeric feedback is transformed into a numeric reward. Otherwise, the reward is the same as the listener's numeric feedback.

Figure 1. DI-C's Model-Based Architecture

Related Work

Data science research on recommender systems in IS and other related fields has focused almost exclusively on induction (learning) of users' preferences from *historical data* (e.g., Adomavicius and Kwon, 2014; Adomavicius et al. 2011; Adomavicius and Zhang, 2012; Ghoshal and Sarkar 2014; Prawesh and Padmanabhan 2014). Most prior work on modeling preferences for playlist generation in particular focused on listeners' preferences for individual songs or artists, irrespective of the order in which they are played; consequently, this work focused on identifying songs or artists that are similar to those to which the listener previously indicated a preference (e.g., Aizenberg et al. 2012; Platt 2003; Wang et al. 2013; Weston et al. 2011). Importantly, the research outlined above is distinguished from our work in that it uses historical data to induce users' preference patterns; in particular, these papers did not attempt to perform both online learning of individuals' sequential preferences from data acquired online (in real time) and to adapt online to these preferences.

Prior research on music perception has established that humans perceive music sequentially (Palmer 2005), and that the pleasure from music is also affected by the sequence of songs (Kaji et al. 2005). Most research on producing playlist recommendations has either considered learning of listeners' preferences for sequences from large historical data, or did not learn preferences over transitions at all, but rather assumed what sequences are desirable for all listeners in all contexts. For example, several works on learning of preferences addressed playlist prediction from large historical data sets (e.g., Chen et al. 2012; Harir et al. 2012; Maillet et al. 2009; McFee and Lanckriet 2011; Natarajan et al. 2013; Zheleva et al. 2010). Other works, including Zheleva et al. (2010), did not attempt to learn sequences from listeners' playlists at all, but made assumptions about what makes sequences enjoyable across contexts. Zheleva et al., for example, place a bound on the distance between two subsequent songs. Natarajan et al. (2013) generalize this approach to explicitly model one-step transitions.

Several recent works considered online adaptation to listeners' music preferences, but addressed meaningfully different problems and settings than those we address here. For example, Chi et al. (2010) assume that all songs in the corpus are pre-labeled as belonging to one of four "valence quadrants" by human annotators who were trained for this task. However, this approach is inapplicable for streaming services more broadly. Chi et al. and King and Imbrasait (2015) use the Q-learning framework (Sutton and Barto 1998) for online adaptation that requires a large number of interactions with listeners before yielding benefits. Thus, for example, King and Imbrasait require multiple days of usage to achieve similar benefits to a random selection of songs. In addition, note that King and Imbrasait also consider settings with a fixed set of songs from a listener's own collection of preferred songs. Hence, it is inapplicable to a setting where the listeners' preferences for songs ought to be discovered simultaneously with transition preferences. Similarly, Shivaswamy and Joachims (2011) recommend articles to readers, but do not aim to learn sequential preferences.

To the best of our knowledge, prior works have not attempted to learn and adapt online to an individual's songs and transition preferences in a given context, during a single listening session. Thus, in this paper we extend our prior work (Liebman, Saar-Tsechansky, and Stone 2015) to study the key design properties that contribute to fast and effective real-time adaptation to listeners' transition preferences from limited feedback, acquired during a single session. Specifically, we study the benefits of the listener reward model we propose toward the ability to aggressively generalize in real time the listener's preferences from a smaller number of interactions with the listener; further, we evaluate the effectiveness of our approach's planning in trading off the immediate and long-term rewards from selecting the next song. In addition, we develop and evaluate two extensions of DJ-MC that account for song costs (royalty fees) when producing a playlist. Finally, we evaluate the sensitivity of the DJ-MC framework to different parameter settings. Overall, prior work has not considered adaptation through interactions in a single session, hence it did not explore the design properties that are beneficial to do so effectively, or whether adaptation to listeners' preferences for song sequences yields any better listening experiences for listeners. This paper aims to bridge these gaps.

Defining Our Playlist Adaptation Problem

We consider a finite set of musical tracks (songs) $M = \{a_1, a_2, \ldots, a_n\}$ from which playlists can be constructed, and assume that playlists include an arbitrary number of songs K. As discussed above, for a reinforcement learning agent to perform well, it ought to effectively predict the consequences of its actions, such as the immediate utility from its action. At any given time, the ordered sequence of songs played thus far, $s \in S$ is known, where S is the set of all possible sequences $S = \{(a_1, a_2, \ldots, a_i) | 1 \le i \le k; \forall j \le i, a_j \in M\}$. At each step k the recommender agent selects the next song to play, $a_k \in M$. After each song is played, the listener provides *feedback*, which captures the immediate utility or enjoyment the listener derives from listening to song a after listening to sequence s. As we will see below, DJ-MC can accommodate different types of feedback by the listener, but requires that the feedback be transformed into a real number *reward*. If the listener's feedback is a real number, such as a score, DJ-MC uses the score itself as the *reward*. Otherwise, when the listener's feedback is like/dislike signals, it is represented by DJ-MC as a +1 and 0 rewards, respectively. Henceforth, we use *feedback* to refer to the signal provided by the listener, and *reward* to refer to DJ-MC's numeric representation of the listener's feedback.

Table 1 outlines the notations used throughout the paper.

DJ-MC *Architecture: Reinforcement-Based Adaptation to a Listener's Preferences*

Our agent's goal is to find and improve over time a policy that specifies the selection of the next song to play, so as to produce the sequence of songs that would be most pleasing to the listener. Because reinforcement learning considers a sequence of actions at any given time, a policy aims to select an action that may not yield the highest immediate reward, but the highest long-term value. Different from the immediate reward received from an (s, a) pair—taking action a after listening to sequence s—the *value* of an (s, a) pair is an estimate of the long-term benefits, that is, the expected cumulative future rewards following a (s, a) pair, when following the same policy π thereafter. Nevertheless, note that our ability to predict the immediate reward is fundamental to accurately estimating the value of an (s, a) pair.

Our first design choice pertains to our agent's architecture suitable for our setting, where adaptation of the playlist to the listener's preferences ought to rely on limited experiences with the listener acquired during a single session. There are two classes of reinforcement learning methods which differ in how they assess the value of state-action pairs to produce a policy (i.e., decide what actions to select). *Model-free* methods learn the value of taking an action a when at state s directly, by collecting *actual* experiences of taking action a at state s, and updating the values of actions only when taking them in the real task. Consequently, model-free methods require significant experiential data and visit the same states many times to converge on good estimates of the value of state–action pairs. However, experiential data in our setting (i.e., playing songs and obtaining the listener's feedback) is particularly scarce and therefore model-free methods are not suitable for our setting. The alternative, *model-based* methods, involve modeling the environment, that is, model a mapping between any given state and action pair (a, s) onto the reward v, in our setting. In this paper, we refer to this mapping as the listener's reward model. The listener's reward model allows the agent to estimate the value of alternative courses of action via simulating experiential data, rather than by relying on actual experiences. The use of models to simulate and assess the value of alternative courses of action is referred to as *planning*. Because experiential data is scarce in our online adaptation setting, in order to allow visiting the same state frequently, our framework adopts model-based learning and planning.

As discussed above, key to DJ-MC's ability to perform model-based planning is to have an internal, listener's reward model so as to allow predict the observed reward after a song is played in any given context. The reward model maps information from the current state and the next song selected onto

Table 1. Key Notations and Terminology

Notation	Description
M	Song corpus: set containing all songs
K	Number of songs in the playlist
k	Counter index of the song number along a playlist: $k = 1, 2, \ldots, K$
a_k	The kth song in a sequence of songs, k is the step/song number in the sequence: $k = 1, 2, \ldots, K$
u	listener
S, s	S: Set of all states, s: a state
P	Transition function
R	Listener reward model
T	Set of terminal states (playlist of K songs)
R_s	Song reward component of reward model R
R_t	Transition reward component of reward model R
Feature	Properties used to describe songs (e.g., loudness to describe a song).
F_s	Set of all song features
Song descriptor	Song features are quantized into $nbin_s$-percentile binary descriptors ($nbin_s$ = 10 for the main results reported here), such that the value of a given song features (e.g., loudness) for a given song is captured by a vector of $nbin_s$-binary descriptors with the value 1 in a single entry that corresponds to the song percentile descriptor's value, and 0 in all other entries (see Figure 3 for illustration)
Transition descriptor	A transition descriptor is a binary indicator corresponding to a transition from one percentile song descriptor to another percentile song descriptor for the same feature (e.g., from 10th percentile to 5th percentile for loudness) (see Figure 3 for illustration)
θ_s	Vector of song descriptors
ϕ_s	Vector of song descriptor weights
θ_t	Vector of transition descriptors
ϕ_t	Vector of transition descriptor weights
$nbins_s$	Number of percentile bins per song feature (song descriptor dimensions)
$\varphi_s^{d,1}$	Weight in ϕ_s corresponding to the lth 10-percentile bin descriptor of song feature d: $\varphi_s^{d,1}, \varphi_s^{d,2}, \ldots, \varphi_s^{d,nbins_s}$
$\varphi_t^{d,1,g}$	Weight in corresponding to a transition descriptor for a transition from the lth binary song descriptor to the gth binary song descriptor of song feature d
z_s	Number of favorite songs polled during initialization
z_t	Number of favorite transitions polled during initialization
Feedback	Listener's feedback provided by the listener and reflects the listener's benefit/enjoyment from the recent song played
v_k	Reward after playing song a_k; reward is a real number representation of the listener's feedback
$\bar{v}_k$	Mean of rewards for songs $a_1, \ldots, a_k$
v_{incr}^k	Value increment used for reward model update after playing song a_k
w_s	Update fraction for song descriptor weights
w_t	Update fraction for transition descriptor weights
q	Planning horizon: Number of songs in the playlist used for planning
M^*	A subset of B percent of songs in the corpus M, with the highest song rewards R_s
B	Percent of songs (used to select subset M^*)
D	Total number of basic features
H	$H \subset M^*$, representative sample of songs drawn from M^* via the δ-*medoids algorithm*
C_s	Song cost

the subsequent reward. As such, the listener's reward model aims to capture the individual listener's preferences. Figure 1 illustrates the architecture of our model-based RL: at each step, DJ-MC observes the listener's feedback, transforms it into a reward v, updates its listener's reward model based on the new reward signal v and observed state s, performs planning using the updated model, and selects the next song (i.e., selects an action).

DJ-MC *Modeling of Listener Reward*

As noted above and illustrated in Figure 1, the listener's reward model is learned from real experiences with the listener. Hence, a fundamental challenge is for the reward model to capture effective patterns of preference that give rise to different rewards in different contexts, but also to ensure that learning these patterns is possible when relying on limited experiences with the listener acquired during a listening session. These considerations are central to the design choices we discuss here.

Toward that end, we propose to model the reward in such a way that generalizes the experiences with the listener aggressively, thereby allowing DJ-MC to use a small number of experiences with the listener to predict the reward from songs played in different contexts that are *similar* to experiences the listener actually encountered earlier. To appreciate why aggressive generalization from limited experiences is essential in our setting, note that for a modest music corpus of 10^3 songs and playlist horizons of only 10 songs, the size of the state space (of possible song sequences) is 10^{30}. It is therefore infeasible to actually try out any meaningful portion of the space, let alone learn the listener's utility for each possible song and transition. Consequently, our goal is to generalize preferences across *similar* songs and states that have been encountered earlier in the listening session.

To achieve particularly aggressive generalization from limited experiences, our framework includes two complementary elements. The first is a factored and compact data representation of both songs and song transitions. The second element is to use a linear model to capture listeners' preferences over the representation of songs and transitions.

Representation of Songs and Song Transitions

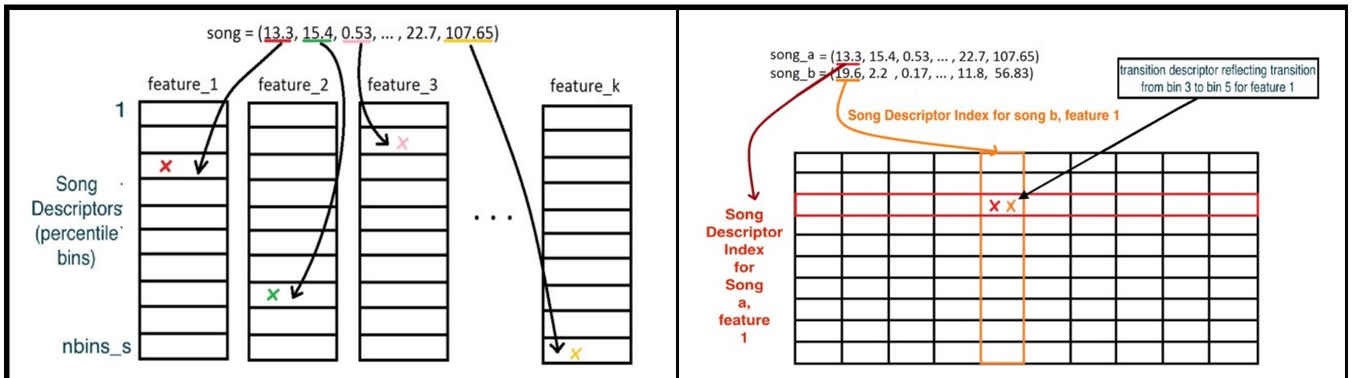
The first element by which we propose to facilitate generalization of preferences from limited feedback acquired during a listening session is a factored and compact representation of songs and song transitions. In particular, we propose to represent a song by a simple vector of song *features*. Consequently, our approach will perform learning of preferences over song features, rather than over individual songs. Because songs can share similar features, this facilitates fast generalization from actual interactions with the listener. In the empirical evaluations that follow, songs are represented by 34 song features corresponding to common spectral auditory properties of a song, such as its rhythmic characteristics, overall loudness, and how these characteristics change over time.[2]

To further expedite generalization from limited experiences with the listener, we also employ a sparse representation of each song feature. The representations of songs aims to be informative while facilitating aggressive generalization from songs encountered in actual experiences onto *similar* ones. Specifically, as illustrated in Figure 2(a), rather than represent the point-value of a given feature for a given song, song features are quantized into $nbins_s$-percentile bin descriptors, based on statistics collected over the complete music database. (Given our data contains 34 features, each represented by a 10-percentile binary descriptors, this results in 340 song descriptors.) Toward the main results reported in this paper, we used $nbins_s = 10$. Thus, the value of a given song feature (e.g., loudness) for a given song is captured by a vector of 10 binary descriptors with the value 1 in a single descriptor that corresponds to the song 10-percentile feature value, and 0 in all other entries. The binned representation of features allows aggressively generalizing from experiences across songs with *similar* (not identical) feature values. Below we will revisit how this representation of song features is operationalized to represent song transitions.

DJ-MC*'s Internal Model of a Listener's Preferences*

DJ-MC's internal listener reward model aims to predict a listener's preferences and is thus a mapping between any song played at any given time onto the reward DJ-MC is likely to receive. Perhaps the most critical design choice we propose to support fast generalization of this model from actual experiences is the kind of mappings we consider for this model.

[2]Note that while the particular features ought to be informative to enable learning, they are independent of the framework we develop to learn and adapt to the listener's preferences. Consequently, our DJ-MC framework can apply with different and potentially more informative features.


song = (13.3, 15.4, 0.53, ... , 22.7, 107.65)
feature_1
feature_2
feature_3
feature_k
1
Song Descriptors (percentile bins)
nbins_s
song_a = (13.3, 15.4, 0.53, ... , 22.7, 107.65)
song_b = (19.6, 2.2 , 0.17, ... , 11.8, 56.83)
transition descriptor reflecting transition from bin 3 to bin 5 for feature 1
Song Descriptor Index for song b, feature 1
Song Descriptor Index for Song a, feature 1

(a) Song feature representation: Each song feature is represented by *nbins*$_s$ descriptors, each of which is a binary percentile bin, indicating whether or not the feature value maps to the corresponding percentile

(b) Song transition representation. Matrix of transition descriptors corresponding to a transition from one percentile bin descriptor to another for a given feature *l*.

Figure 2. Illustration of Song Percentile Bin Descriptions (a) and Transition Descriptors (b)

Despite a rich literature on the psychology of human musical perception (Tan et al. 2010), there is no canonical model of the human listening experience to be used for our listener reward model. We thus lean and update a listener preference model from the data acquired during a listening session. In particular, we propose to represent listeners' preferences by a simple, linear model such that fewer parameters must be learned, thereby reducing the risk of over-fitting. The relationship between model complexity and the amount of training data required for producing accurate predictions is explained by bias–variance tradeoff in data-driven learning (Hastie et al. 2009). In particular, when induced from small amounts of training data, complex models tend to exhibit high estimation variance and thereby larger estimation (prediction) error; this is because a limited training data set is unlikely to be representative of the rich set of patterns that can be captured by the (complex) model, and thus a highly expressive, complex model tends to overfit the atypical patterns in the data (Hastie et al. 2009). By contrast, simpler models tend to generalize better from small training sets because of their limited ability to fit the training data well. We revisit and empirically validate this design choice in the evaluation section.

To yield a simple representation of listeners' preferences we model a listener's reward $R(s, a)$ as the sum of two components: (1) the listener's preference over songs, R_s: $A \rightarrow \mathbb{R}$, and (2) the listener's preference over transitions from the sequence of songs thus far onto the new (next) song, R_t: $S \times A \rightarrow \mathbb{R}$. Thus: $R(s, a) = R_s(a) + R_t(s, a)$.

We define the rewards from playing the current song, R_s, as a linear function of the song's percentile bin descriptors: each song descriptor contributes independently to the listener's utility from the song. In this linear representation, each song descriptor is associated with a corresponding weight, reflecting the pleasure the listener derives from songs with that descriptor active, giving rise to a weight vector for listener u denoted by $\phi_s(u)$. The reward from playing an individual song a for listener u is thus given by $R_s(a) = \phi_s(u) \cdot \theta_s(a)$, where $\theta_s(a)$ denotes the vector capturing all the descriptors for song a. Thus, the weight vector $\phi_s(u)$, is learned separately for each individual listener u, and captures the listener's song preferences (i.e., how different descriptors contribute to the listener's reward).

A main premise of this work is that a listener's pleasure also depends on the *sequence* in which songs are played, and we will later explore whether listeners' experiences can be improved if these preferences are learned and adapted to. Following the discussion above, we represent a listener's preferences for song transitions via a simple linear mapping so as to render generalization effective, given limited experiences with listeners during a listening session.

We capture a listener's pleasure from a sequence by representing preferences as depending on the transition from the complete sequence listened to so far, onto the current song. In particular, at a given state s, representing the sequence of songs played thus far $(a_1, \ldots, a_{k-1})$, the reward derived by the listener from transitioning to a new song a_k is represented by the sum of utilities derived from the transition between each of the songs played thus far in the sequence and the current song, $a_k : R_t(s, a_k) = R_t((a_1, \ldots, a_{k-1}), a_k) = \Sigma_{i=1}^{k-1} \frac{1}{i^2} r_t\left(a_{k-i}, a_k\right)$, where $r_t(a_i. a_j)$ denotes the listener's utility from listening to song a_j sometime *after* having listened to song a_i. Importantly, in the expression above, a song a_{k-1} (played i songs earlier in the sequence) has a probability of $\frac{1}{i}$ of affecting the transition reward (i.e., being "remembered"); in addition, a song's impact on the reward from playing the next song decays over time, and is reflected by a second factor of $\frac{1}{i}$ (thereby yielding the term $\frac{1}{i^2}$). Having a decaying effect on the reward by songs played earlier in the sequence is inspired by known properties of human perception (i.e., by the stochastic dependence on remembering earlier events, and evidence of working memory having greater emphasis on the present) (Berz 1995; Davies 1978; Tan et al. 2010).

To facilitate generalization of transition preferences from limited training experiences, we use a simple linear form to represent the utility from a transition, $r_t(a_i. a_j)$. In particular, for each individual listener, the utility from a transition, $r_t(a_i. a_j)$, is modeled as a linear function: $r_t(a_i. a_j) = \phi_t(u) \cdot \theta_t(a_i. a_j)$, where θ_t is a binary vector of the transition descriptors, and $\phi_t(u)$ is a listener-specific weight vector reflecting preferences over the transition descriptors. The transition preferences parameters of $\phi_t(u)$ are learned afresh when adapting to an individual listener in a given context.

Similar to our representation of songs discussed above, to further facilitate generalization, the descriptors used to represent transitions follow the same compact representation used to represent song descriptors. Specifically, as illustrated in Figure 2(b), for each auditory song feature (e.g., loudness), the corresponding transition descriptor matrix θ_t captures a transition between one 10-percentile bin descriptor for this auditory feature to another. Thus, given $nbins_s$ descriptors used to represent each song feature, there are $nbins_s^2$ transition descriptors representing the transition for each song feature. In the empirical evaluations that follow, for each feature, there are 100 (10 × 10) transition descriptors, each reflecting the possible transition between one 10-bin song descriptor to another of the same feature, and where only one has value 1. As in the case of songs, our use of binary values over bins instead of real-values allows DJ-MC to more aggressively generalize preferences across similar transitions.

Finally, recall that our representation of transitions is linear with few parameters, hence it does not capture other possible dependencies, such as the joint dependence of a listener's pleasure from transitions between different song descriptors (such as transitions between a certain percentile of loudness and a percentile of pitch dominance). Yet, such dependencies are likely to be informative. As discussed earlier and explored empirically later on, a complex reward model, even if it can better approximate the true complexities underlying listeners' preferences, is unlikely to be learned effectively from a limited number of actual experiences with the listeners. We will later examine this question empirically.

Reinforcement Learning of a Playlist Generation Policy

In this section, we discuss DJ-MC's reinforcement learning so as to adaptively personalize a listener's playlist while interacting with the listener. The DJ-MC design includes two major elements: online *learning* of the listener reward model parameters (ϕ_s and ϕ_t) and then *planning* a sequence of songs to play based on the learned preferences.

DJ-MC begins with an initialization of the listener reward parameters before learned preferences are available for playlist planning and production. Initialization occurs only once, at the beginning of the session. As we will see below, in simulation we let users specify an initial list of songs they enjoy. We later show that this initialization step also can be replaced by random exploration by DJ-MC to yield effective adaptation to listeners. The next phase, learning on the fly, enables DJ-MC to sequentially update and improve its listener reward model parameters after each interaction with the listener. *Planning* of the playlist based on the learned preferences corresponds to the selection of the *next* song to play in the sequence at each step. Below we describe DJ-MC's initialization, learning on-the-fly algorithm, and the planning for playlist generation. The complete learning pseudocode is outlined in Algorithm 5.

Initialization for Song and Transitions Preferences

Recall that the weight vectors for song and transition descriptors, $\phi_s(u)$ and $\phi_t(u)$, aim to capture the listener u's song and transition preferences, and it is used to guide DJ-MC's interactions with the listener. As is the case in any online learning framework, it is necessary to determine how the agent interacts initially with its environment; in our setting, we initialize the model to determine what songs DJ-MC initially plays and on which it receives the listener's feedback during its initial interactions with the listener. In this section, we describe how the listener's reward model is initialized in the simulation studies; in the empirical evaluations we describe a simpler form of initialization that was used in the experiments with human listeners.

The reward model's song weights are first initialized such that the weights corresponding to all descriptors (percentile bins) for the same feature (e.g., loudness) are uniform and sum to

Algorithm 1. Initialize Song Preferences R_s

```
Input  M: Song corpus to play to listeners,
       $M_\theta$: All songs in M, each song $a_i$ represented via a vector of song descriptors $\theta_s(a_i)$
       $z_s$: Number of preferred songs to be provided by listener
       $nbins_s$: Number of percentile bins per song feature
1.  Initialize all song weights in $\phi_s$ to $\frac{i}{z_s + nbins_s}$
2.  $preferredSet = \{a_1, \ldots, a_{z_s}\}$ (preferred songs chosen by the listener from M)
3.  for $i = 1$ to $z_s$ do:
4.      $\phi_s = \phi_s + \frac{1}{(z_s + nbins_s)} \bullet \theta_s(a_i)$
5.  end for
6.  Return: $\phi_s$
```

1: each element of $\phi_s(u)$ is initialized to $1/(z_s + nbins_s)$. The listener is then asked to select z_s favorite songs from the corpus M. For each favorite song, a, the weight in $\phi_s(u)$ associated with each relevant descriptor bin corresponding to song a is incremented by $1/(z_s + nbins_s)$. Once the initialization is complete, the sum of weights corresponding to all descriptors of song feature d is 1:

$$\forall d \in F_s : \Sigma_{l=1}^{nbins_s} \varphi_s^{d,1} = \frac{z_s + nbins_s}{z_s + nbins_s} = 1$$

where F_s is the set of all song features, and $\varphi_s^{d,1}$ is the weight in ϕ_s corresponding to the lth binary descriptor of song feature d. The pseudocode for DJ-MC's initialization is shown in Algorithm 1.

DJ-MC's initialization for song *transition* preferences, described in Algorithm 2, also begins with a uniform weight for all transition descriptors. Specifically, for each song feature, the weight corresponding to each transition from descriptor (bin) i to descriptor j of the same song feature, denoted by $\varphi_t^{d,i,j}$ is first initialized to $1/(z_t + (nbins_s)^2)$, where z_t is the number of transitions queried and $(nbins_s)^2$ is the number of transition descriptors corresponding to each song feature.

The listener is then polled for transition preferences. Given the large space of possible transitions, to make the choice simpler for the listener, DJ-MC first draws a subset H of songs that represents the variety of the transition space efficiently. Specifically, DJ-MC considers 50% of the songs M^* in corpus M with the highest song rewards R_s based on its reward model, and then draws a representative sample H from M^* via the δ-medoids sampling algorithm (Liebman, Chor, and Stone 2015). The δ-medoids algorithm, designed specifically for music domains, draws a representative sample of songs, such that no song in the corpus is more than a parameter δ away from a sampled song. In any specific setting in which DJ-MC will be applied, the sample size for H can be controlled by the parameter δ. In our simulation experiments, δ was set to be the 10th percentile of the distance histogram between all pairs of songs in M^*. To initialize the listener's transition weights in the reward model, DJ-MC first selects a song uniformly at random from H (line 5), and then sequentially queries the listener z_t times about which song $a_i \in H$ she would like to listen to next, so as to produce a sequence of z_t transitions (lines 6–7). In the simulation studies we identify the song the listener chooses next by simulating the listening experience, including the nondeterministic history-dependent transition reward, and choosing the one with the largest reward. DJ-MC then updates the weight for the selected transition by increasing the corresponding transition weight by $1/(z_t + (nbins_s)^2)$,. Once the initialization is complete, the sum of weights of all transitions from one song percentile (bin) descriptor to another for a given song feature d is 1. Note that this form of initialization is suitable when the listener has some familiarity with the song in H.

Online Learning of Sequential Preferences: Learning-By-Doing

To learn while interacting with the listener during a listening session, DJ-MC plays songs and iteratively updates the corresponding songs and transition parameters ϕ_s and ϕ_t according to the listener's feedback. The pseudocode for DJ-MC's online learning algorithm is presented in Algorithm 3. Specifically, at each step k, song a_k is played, after which DJ-MC observes the listener's feedback. As we note earlier, the listener's feedback, which reflects the listener's enjoyment,

Algorithm 2. Initialize Transition Preferences

```
Input   $M$: Song corpus
        $z_s$: Number of transitions to poll the listener
        $(nbins_s)^2$: Number of transition weights per feature
        $R_s$: Initial song preference weights (obtained by Algorithm 1)
1.   Initialize all transition weights in $\phi_t$ to $1/(z_t + (nbins_s)^2)$
2.   Select upper median of $M$, $M^*$, based on $R_s$
3.   $\delta$ = 10th percentile of all pairwise distances between songs in $M$
4.   Representative set of songs $H = \delta$-medoids $(M^*)$
5.   $song_0 \leftarrow$ draw a song uniformly at random from $H$
6.   for $i = 1$ to $z_t$ do (query the listener which song she likes to listen to next)
7.       $song_0 \leftarrow$ chosen by the listener from $H$
8.       $\phi_t = \phi_t + \frac{1}{z_t + (nbins_s)^2} \bullet \theta_t(song_{i-1}, song_i)$
9.   end for
10.  Return: $\phi_s$
```

can take different forms, such as a real number score or a like/dislike signal, which is later mapped to a real number, reward. In the evaluations below, when the listener feedback is a real number, it is also used as the corresponding reward v. Otherwise, in settings where the listener provides feedback in the form of like/dislike signal, this feedback is mapped to a numeric reward of value +1 or 0, respectively.

After each song a_k is played, two elements determine the magnitude and direction of change in the reward model parameters corresponding to the a_k's song and transition descriptors. First, we aim to use listener feedback to revise the weights of the model in the direction of the feedback signal. Such an update aims to increase weights corresponding to song and transition descriptors that the user marked as favorable (positive feedback), and decrease weights corresponding to song and transition descriptors of songs and transitions that the user did not find favorable (negative feedback). Further, because we aim to generalize from feedback quickly, it is desirable for the update to be more aggressive the more unusual the reward is (i.e., the farther the signal is from the reward expectation). Assuming the average of rewards thus far $\overline{v}_{k-1} = \frac{\Sigma_1^{k-1} v_i}{(k-1)}$ is the maximum likelihood estimate for the reward expectation, and the reward for the recently played song is v_k, the ratio $\frac{v_k}{\overline{v}_{k-1}}$ captures how different the recent reward is relative to the maximum likelihood reward expectation. Formally, we define the reward increment after receiving reward v_k as $v_{inc}^k = \log\left(\frac{v_k}{\overline{v}}{}_{k-1}\right)$, thereby capturing both the *direction* and *magnitude* of the recent reward relative to expectation. Thus, the incremental reward v_{inc}^k is negative if $v_k < \overline{v}_{k-1}$ and positive otherwise; similarly, the reward increment magnitude is greater the farther the reward v_k is from the average reward $\overline{v}_{k-1}$. Consequently, a song is less likely to be played if the song and resulting transition are similar to songs and transitions played previously and which received (larger) negative rewards.

The second element determining how parameters of the reward model are updated is the proportion of the listener reward attributed to either songs or transitions. Specifically, it is possible that a listener provides a single feedback reflecting her overall enjoyment from the recent song and transition together; thus, DJ-MC will aim to attribute a proportion of this reward to the song being played and the remaining proportion to the enjoyment from the transition from the sequence thus far onto the current song. The proportion of reward DJ-MC attributes to the song and transition is proportional to the ratio between song and transition reward to the total reward, as predicted by DJ-MC's internal listener reward model, as in a maximum likelihood estimate. Concretely, let $R_s(a_k)$ and $R_t(a_{k-1}, a_k)$ be the expected song and transition rewards yielded by DJ-MC's reward model, respectively. Thus, the update of weights capturing the proportion of reward attributed to the song and transition is given by $w_s = \frac{R_s(a_k)}{R_s(a_k)+R_t(a_{k-1},a_k)}$ and $w_t = \frac{R_t(a_{k-1},a_k)}{R_s(a_k)+R_t(a_{k-1},a_k)}$, respectively.

Finally, the model updates (lines 5 and 6 in Algorithm 3) employ the common practice of updating the model with an

Algorithm 3. Model Update

```
Input   Song $a_k$
        $k$: step/song number in the sequence
        $v_k$: reward received after playing $a_k$
        $\bar{v}_k = average(\{v_1, \ldots, v_{k-1}\})$
        Current reward model $R$ (with parameters $\phi_s$, $\phi_t$)
1.  $v_{inc}^{k} = \log\left(v_k / \bar{v}_k\right)$
    Reward model update:
2.  $w_s = \frac{R_s(a_k)}{R_s(a_k)+R_t(a_{k-1},a_k)}$
3.  $w_t = \frac{R_t(a_{k-1},a_k)}{R_s(a_k)+R_t(a_{k-1},a_k)}$
4.  $\phi_s = \frac{k}{k+1}, \phi_s + \frac{1}{k+1} \bullet \theta_s \bullet w_s \bullet v_{inc}^{k}$
5.  $\phi_t = \frac{k}{k+1}, \phi_t + \frac{1}{k+1} \bullet \theta_t \bullet w_t \bullet v_{inc}^{k}$
6.  For each song and transition feature, normalize the descriptor weights corresponding to each song feature and
    each feature transition, such that they add up to 1, as follows:
    $\forall d \in F_s : \varphi_s^{d,l} = \frac{\varphi_s^{d,1}}{\sum_{i=1}^{nbins_s} \varphi_s^{d,i}}.$    $\forall d \in F_s : \forall d \in F_s, \forall l, g = 1,2,\ldots,nbins_s : \varphi_t^{d,l,g} = \frac{\varphi_t^{d,l,g}}{\sum_{i=1}^{nbins_s} \sum_{j=1}^{nbins_s} \varphi_s^{d,i,j}}$
7.  Return: updated reward model parameters: $\phi_s$, $\phi_\tau$
```

attenuating learning rate proportional to the number of songs played thus far. This procedure guarantees convergence over time (i.e., as $k \rightarrow \infty$: $1/(k+1) \rightarrow 0$ and . However, more importantly, for short playlists considered in this paper, this update step simply yields a continuous change in balance between updating the model based on new and potentially noisy individual feedback (the listener's feedback to the last song played), and how much we trust the current reward model, which already reflects all past experiences with the listener (prior songs played and corresponding feedback). In particular, after each song, the current model weights are reweighted by a factor of $\frac{k}{k+1}$, whereas the update step based on the new feedback is weighted by a factor of $\frac{k}{k+1}$. As a result, in the early learning stages the model update is more heavily affected by new experiences than later on, when more experiences with the listener have been captured by the model. Finally, after each update, the weights for all features are normalized so they sum up to 1.

Planning: Exploring and Assessing the Long-Term Value of Different Songs to Play Next

Because DJ-MC simultaneously *learns* and *acts*, it has two goals for selecting the next song: the first is to *explore*, namely, try out different songs to explore the listener's preferences; a second goal is to *exploit* its current knowledge of the listener's preferences, so as to select a song that will yield a higher cumulative reward.

To explore the listener's preferences, DJ-MC simulates different random sequences, and it exploits its knowledge by using its listener reward model to select the song that yields the highest long-term value. Toward both goals, a variant of Monte Carlo tree search (MCTS) proposed by Urieli and Stone (2013) serves as a particularly suitable heuristic framework for the problem in our setting. Importantly, MCTS is a general framework that allows us to explore the space of actions to take, and which does not require an explicit, evaluation function to assess their values; namely, in our context, it does not require a function that computes in closed-form the (long-term) value of playing a given song next. Thus, to quickly explore alternative courses of action, MCTS is used to execute simulations that yield plausible song sequences, multiple times. These sequences are referred to as Monte Carlo rollouts. DJ-MC then selects the next song that is followed by advantageous sequences in expectation (i.e., rollouts that yield higher cumulative reward as estimated by the current listener reward model).

DJ-MC's playlist planning algorithm is shown in Algorithm 4. For planning, DJ-MC considers a subset of B percent of the

Algorithm 4. DJ-MC Planning via Tree Search

```
Input   Song corpus, $M$
        planning horizon $q$
        Current preference model $R$ (with parameters $\phi_s$, $\phi_t$)
        Currently played song $song_0 \Leftarrow a_k$
        $B$: percent of songs from corpus $M$ to use in planning
1.  Select a set of B percent of songs from $M$, $M^*$, with the highest song reward $R_s$
2.  BestTrajectory = null
3.  HighestExpectedPayoff = $-\infty$
4.  While computational power not exhausted do:
5.      trajectory - []
6.      for $i$ = 1 to q do:
7.          $song_i \leftarrow$ selected randomly from $M^*$ (avoiding repetitions)
8.          add song to trajectory
9.      end for
10.     expectedPayoffForTrajectory = $R_s(song_i) + \sum_{i=2}^{q}\Big(R_t\big((song_1,\ldots,song_{i-1}),song_i\big) + R_s(song_i)\Big) =$
11.     if expectedPayoffForTrajectory > HighestExpectedPayoff then
12.         HighestExpectedPayoff = ExpectedPayoffForTrajectory
13.         BestTrajectory = trajectory
14.     end if
15. end while
16. Return: First song in BestTrajectory
```

songs corpus with the highest song rewards, based on R_s (line 1 in Algorithm 4). This reduction in the song space increases the likelihood of selecting an enjoyable song (although it might not improve the likelihood of identifying an enjoyable sequence and transitions). To identify the next song likely to produce enjoyable sequence thereafter, DJ-MC follows the MCTS framework and simulates possible song sequences. At each step, DJ-MC simulates a trajectory of q songs selected at random from the top B percent of songs with the highest song rewards (lines 6–9) and computes the expected reward (line 10) for the resulting trajectory based on the current listener reward model. Following the general MCTS framework, DJ-MC repeats this process until it is either interrupted by a request for the next song to play, or otherwise has exhausted its computational budget of how many sequences to explore. DJ-MC then identifies the song followed by the trajectory that yields the highest expected reward of those explored thus far (lines 11–14). Importantly, because the listener's preferences as reflected by the reward model continuously improves with more experience with the listener, DJ-MC selects only the first song of the most promising trajectory to be the next song played, rather than using the entire trajectory as the complete playlist. After each step, DJ-MC re-plans the playlist, using an increasingly better listener reward model to produce the remaining playlist. Toward the main results in this paper, the planning horizon (number of songs in the sequence) for the simulated sequences during planning is $q = 10$. In addition, $B = 50\%$ for the main results reported below. We later explore DJ-MC's performance with different values for q and B. Algorithm 5 presents the complete pseudocode for DJ-MC's architecture.

DJ-MC Evaluation

We evaluate DJ-MC's performance both in simulation and in an experiment with human listeners. The evaluations aim to first establish whether DJ-MC's design, particularly its internal listener model, song and song transition representations, and learning and adaptation algorithms, facilitates DJ-MC's adaptation to listeners' transition preferences exclusively based on online experiences. We will also explore whether DJ-MC's simple internal listener model benefits online adaptation relative to an alternative, more complex model that reflects listeners' true preference patterns.

Second, note that the feasibility of learning and adapting to listeners' sequential preferences is also necessary in order to establish whether the adaptation yields better experiences for listeners. This result is important to inform the potential benefit from future research efforts to improve online adapta-

Algorithm 5. The DJ-MC Framework

```
Input  $M$ = song corpus
       $q$ = planning horizon
       $z_s$ = number of songs used for song preference initialization
       $z_t$ = number of songs used for transition preference initialization
       $nbins_s$ = number of percentile bin descriptors per song feature
       $nbins_t$ = number of percentile bin descriptors per transition feature
       $B$:  percent of top songs to use during planning
1.  Call Algorithm 1 (with $M, z_s, nbins_s$) to initialize song weights $\phi_s$
2.  Call Algorithm 2 (with $M, z_t, nbins_t, \phi_s$) to initialize transition weights $\phi_t$
3.  $k = 0$
4.  While listener requesting another song (listening session continues) do:
5.      $k = k + 1$
6.      Select the next song:  $a_k \Leftarrow$ Run Algorithm 4 (with $M, q, R,$ current song playing $song_0 \Leftarrow a_k, B$)
7.      Obtain listener feedback after listening to song $a_k$, represent feedback as real number reward $v_k$, and
        compute average reward thus far $\bar{v}_k$
8.      Update reward model $R$';s parameters:  $\phi_s, \phi_t \Leftarrow$ Run Algorithm 3 (with $a_k, k, v_k, \bar{v}_k, \phi_s, \phi_t$)
9.  End while
```

tion of listener's temporal preferences, and from similar efforts to improve the personalization of other digital services.

Data on songs were derived from the Million Song Dataset—a publicly available collection of audio features and metadata for a million contemporary music tracks. The dataset includes songs by 44,745 different artists and 10^6 different tracks. Recall that our design includes a compact song representation, such that songs are factored as a vector of descriptors, reflecting the spectral fingerprint of the song, its rhythmic characteristics, its overall loudness, and their change over time. In this study, we used common acoustic features available in the Million Song Dataset (Bertin-Mahieux et al. 2011) to extract 12 meta-features, out of which two features are 12-dimensional, resulting in 34 features. The complete set of features is summarized in Table 2. Specifically, the first feature is the 10th percentile of beat durations in the song. Loudness was obtained directly from amplitude. Pitch dominance weights each of the 12 possible pitch classes based on their average presence in a song over time. Timbre weights correspond to the average weights of the 12 basis functions used by the Echo Nest analysis tool to capture the spectro-temporal landscape of the song.

Evaluations in Simulation

We begin with an empirical evaluation of DJ-MC in simulation. To produce simulated listeners, we extracted song transition data from a real playlist archive, collected by Berenzweig et al (2004) that reflects playlists produced by real listeners "in the wild." Berenzweig et al. gathered 29,000 playlists from The Art of the Mix (www.artofthemix.org), a repository for playlist hobbyists. This corpus is appealing for listener modeling because the playlists were generated by individual users, rather than a commercial radio DJ or a recommendation system.

We used the above playlist sources to generate listener models for the simulation studies.[3] To produce listener models, we first produce different listener types by generating 10 playlist clusters via k-means clustering over the playlists, where each playlist is represented as an artist-frequency vector. A listener is generated by first drawing at random the cluster to which the listener belongs, followed by sampling 70% of the song transition pairs in that cluster to represent the listener's songs and transitions preferences. Algorithms 1 and 2 are then applied to the sampled songs and transitions to fit the listener's reward model. This process yields simulated listeners with linear preference patterns as those used internally by DJ-MC to model listeners.

The feedback provided to DJ-MC by a simulated listener is a (real number) score, produced by the simulated listener's reward model.

Because prior work had not proposed a general framework for adaptation of playlist transition preferences during a listening session to fit a listeners' preference in a given current context,

[3]Recall, that the representation of songs (e.g., the spectral properties, etc.) is available from the Million Songs dataset. Thus, for the construction of simulated listeners, we used a playlist only if the Million Songs dataset includes a representation for at least one pair of consecutive songs in the playlist.

Table 2. Features Used for Song Representation (Tempo Data is based on beat durations)

Features	Features Indices
10th and 90th percentiles of tempo	1, 2
Average and variance of tempo	3, 4
10th and 90th percentiles of loudness	5, 6
Average and variance of loudness	7, 8
Pitch dominance	9–20
Variance of pitch dominance	21
Average timbre weights	22–33
Variance in timbre	34

in the empirical evaluations that follow we aim to establish DJ-MC's ability to learn listeners' playlist preferences from online experiences, and we evaluate DJ-MC's robustness when listeners have different and more complex preference patterns than those assumed internally by DJ-MC. We also aim to assess whether DJ-MC's account for sequential preferences can improve listeners' experience relative to an approach that learns and adapts the playlist to include songs the listener enjoys the most, irrespective of their order. We will then explore the robustness of DJ-MC's simple listener reward model by evaluating whether it yields better playlists than an alternative design that brings to bear complex patterns of preferences.

To establish whether adapting to a listener's sequential preferences is beneficial we compare DJ-MC with two alternative baselines: a GREEDY, adaptive approach that learns the listener's preferences online as well, but that chooses to play the song with the highest song reward, and a RANDOM approach. Consequently, because it does not account for transition preferences, the GREEDY alternative may not play a song that is less desirable to a listener in order to produce a more desirable sequence. We compare both approaches to a benchmark that simply chooses songs uniformly at random.

As noted in the introduction, because songs are the primary factor that affect listeners' experiences and their enjoyment, GREEDY is expected to yield high *song* rewards for listeners. We will therefore aim to evaluate whether DJ-MC is able to capture listeners' sequential preferences in addition to song preferences, and whether these differential preferences yield a statistically significant improvement in the listener's experience over the GREEDY approach.

Finally, in each experiment, a 1000-song corpus was drawn at random from the Million Song Dataset to serve as the song corpus. For initializing song and transition rewards we use Algorithms 1 and 2, and poll listeners for 10 songs ($z_s = 10$) and 10 transitions ($z_t = 10$). For Algorithm 4, we set a planning horizon of $q = 10$, and a computational budget of 10,000 rollouts. In the experiments with human listeners, discussed below, the time between songs was used to produce and evaluate rollouts.

Results

Figure 3 presents the reward histogram obtained by the DJ-MC, GREEDY, and RANDOM approaches after 10 and 30 iterative steps, where a step refers to one pass in the loop beginning in line 5 of Algorithm 5 (i.e., selecting and playing a song, receiving the listener's feedback, and updating the listener model). As shown, DJ-MC achieves significantly better rewards for users than both the GREEDY and the RANDOM playlists and the differences are statistically significant at $p << 0.01$. Specifically, DJ-MC yields higher rewards after 10 steps, demonstrating that it is able to learn early on differential preference patterns than those exploited by GREEDY, which does not capture sequential preferences; these results also establish that DJ-MC effectively plans desirable song sequences based on these differential preferences.

A Feature-Dependent DJ-MC

In this section, we aim to establish whether having a simple linear reward model is indeed more beneficial toward generalization from limited experiences than a more complex model. The question we aim to answer next is whether DJ-MC's performance is undermined relative to its performance with a listener reward model that reflects the true complex patterns underlying listeners' preferences, and, if so, how much is lost.

In the next experiment, simulated listeners are such that their reward is derived from the joint distributions of pairs of fea-

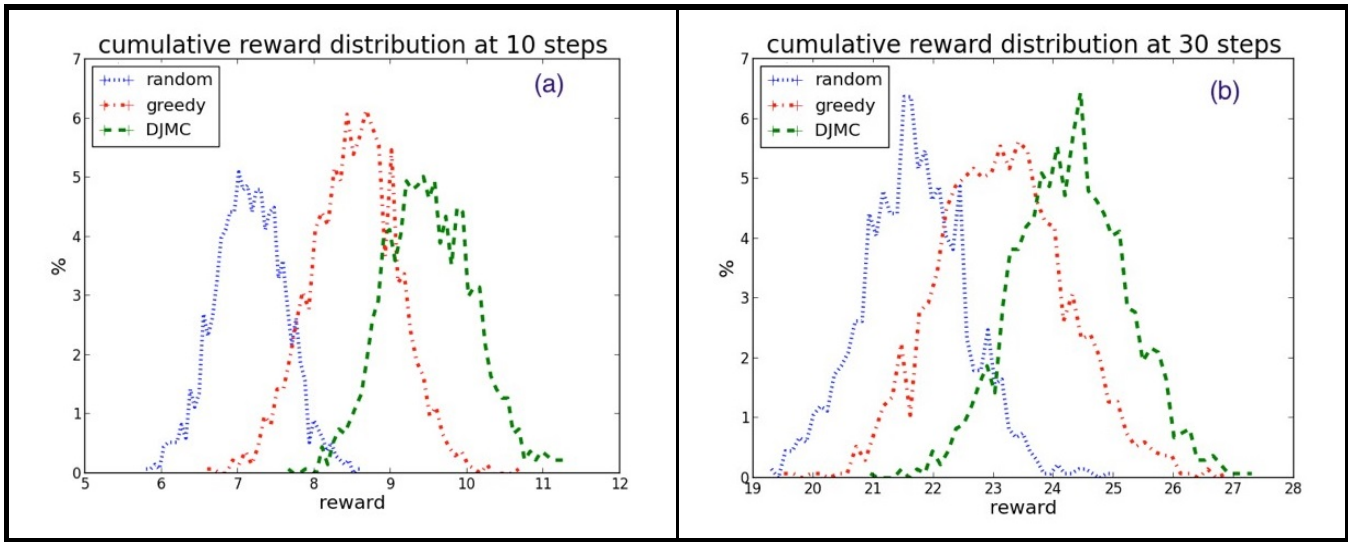


Figure 3. Cumulative Reward Histogram after Playing 10 (a) and 30 (b) Steps, with Listeners Based on Real Playlist Data (DJ-MC outperforms the RANDOM and greedy approaches $p << 0.01$)

tures. As before, we let a listener's reward model R be given by $R(s, a) = R_s(a) + R_t(s, a)$. However, a listener's enjoyment is mapped from the joint distribution of each feature-pair, in a 10 × 10 grid partitioned by the 10th percentiles of each feature. Enjoyment from the transitions is similarly represented by the transitions between feature-pair bins. Thus, the indicator and weight vectors ϕ_t, θ_t are defined over feature pairs rather than over independent features. Formally, $R_s(a) = \phi_s(u) \cdot \theta_s(a)$ and $R_t(s, a_n) = R_t((a_1, \ldots, a_{n-i}), a_n) = \sum_{i=1}^{n-1} \frac{1}{j^2} r_2\left(a_{n-i}, a_n\right)$ with $r_t(a_i, a_j) = \phi_t(u) \cdot \theta_s(a_i, a_j)$ as before. Simulated, feature-dependent listeners were generated by drawing 30 pairs of features at random, while ensuring that each of the original 34 features are selected at least once. Subsequently, (30 × 10 × 10) song weights are assigned to the 100 bins per feature pair, reflecting a random draw of a listener's preferences for songs. Similarly, $30 \times 10^2 \times 10^2$ transition weights are assigned to the 10,000 bins per feature-pair transition, reflecting the corresponding listener's preferences for song transitions.

We now compare DJ-MC's performance to that of a DJ-MC variant, DJ-MC-DEPENDENCY, with a listener reward model structure that matches the preferences of the more complex, simulated listeners. Specifically, DJ-MC-DEPENDENCY's reward model matches the pairwise feature dependencies used to represent the reward of the simulated listeners: DJ-MC-DEPENDENCY thus learns 10 × 10 weights per feature pair and $10^2 \times 10^2$ weights per feature pair transition. Because it is unknown which k feature pairs are active for a given listener, DJ-MC-DEPENDENCY learns the complete set of (561) feature pairs. The reward histograms are presented in Figure 4.[4] As shown, DJ-MC outperforms DJ-MC-DEPENDENCY and both DJ-MC variants outperform the GREEDY and the RANDOM alternatives. These results establish that DJ-MC's simple, linear reward model contributes to its robust performance and to its ability to generalize from limited experiences with the listener. DJ-MC-DEPENDENCY's attempt to learn a complex model from very limited number of experiences leads to overfitting of its interaction with the listener, thereby undermining DJ-MC-DEPENDENCY's planning of song sequences. Because DJ-MC generalizes *simple* predictive patterns of listeners' sequential preferences, it is able to achieve better prediction of rewards sooner, within 10 learning experiences. A simple representation of listeners' preferences is thus instrumental to effective learning of and adaptation to a listener's contextual preferences in real time.

Evaluation with Human Listeners

We now evaluate DJ-MC's performance with human listeners whose preferences are unknown, potentially heterogeneous, and can be arbitrarily complex. In this experiment, we explore two questions. First, we aim to establish whether the listener's experience is affected at all by sequences: if human listeners' experiences are unaffected by the sequence in which

[4]Note that the feature-dependent reward values are smaller than the rewards of the simpler model reported earlier. This is an artifact of scaling the model weights so that they sum up to 1.

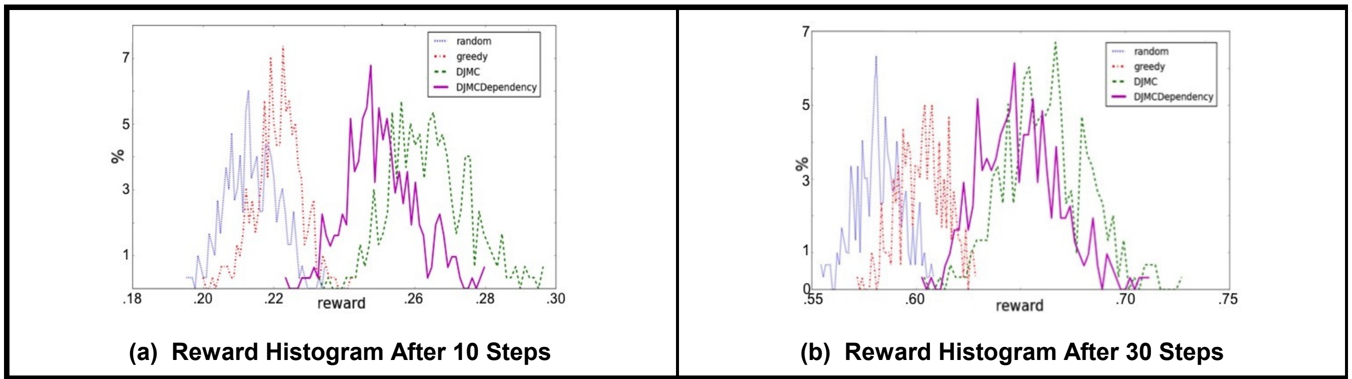


Figure 4. Cumulative Reward Histograms After 10 and 30 Steps, with Feature-Dependent Listeners (DJ-MC yields higher rewards ($p << 0.01$))

songs are played, we expect listeners to attribute random rewards o different transitions, and it would not be possible to induce predictive patterns for transition preferences. By contrast, if sequences do matter and if DJ-MC is able to learn these preference and use them effectively to plan playlists, then DJ-MC will exhibit better listener rewards. Should DJ-MC's design enable adaptation to a human's unknown and arbitrarily complex sequential preferences, this will also offer further evidence of the robustness of DJ-MC's design. Second, the experiment with human listeners also aims to explore the transition and song rewards to understand whether DJ-MC trades off between playing songs a listener enjoys and choosing songs that yield enjoyable transitions.

We evaluated DJ-MC in a lab experiment with 47 graduate students at the McCombs School of Business at the University of Texas at Austin. The song corpus included songs from the Million Song Dataset that also appeared in the *Rolling Stone Magazine*'s list of 500 greatest albums of all time.[5] Each participant interacted with a playlist generator that was based on either DJ-MC or GREEDY. The participants were randomly assigned into one of two groups: 24 participants interacted with the GREEDY approach and the remaining 23 interacted with DJ-MC. Each session began with the RANDOM approach, playing 25 songs drawn at random.

To keep the duration of the experiment reasonable, each song was played for 60 seconds before transitioning (with a cross-fade) to the next song.[6] After each song, participants were asked whether they liked or disliked the song as well as the transition to it, and provided this feedback by clicking on a like/dislike button, in a graphic user interface. Before the experiment began, each participant received individual guidance on how to use the interface and practiced using the interface to provide feedback on several songs to ensure correct usage of the interface. The listener's feedback of a "like" signal for either transition or song was represented by DJ-MC as a reward, and "dislike" feedback was represented as a 0 reward.

As noted above, listeners provide *separate* feedback signals for song and song transition enjoyment in order to later explore tradeoffs done by DJ-MC between satisfying listeners' song and transition preferences. However, as before, DJ-MC receives only a single, unified reward, representing the listener's overall enjoyment. In the simulation studies, initialization was done following Algorithms 1 and 2, where the listener is polled for 10 songs she enjoys; however, initialization can also be done by initially exploring the space of songs and transitions uniformly at random. Indeed, to expedite the experiments in the lab with human listeners, once the listener's reward model weights were set to a uniform value (indicating all songs and transitions are equally enjoyable), initialization of songs and transitions were done by playing a sequence of 25 songs drawn uniformly at random, and updating the listener model based on the listener's feedback after each song. Online learning and adaptation of the learned model then followed for an additional 25 songs. Finally, the Monte Carlo rollouts and their evaluations for selecting the next song were done during the time the current song was being played, without imposing the computational budget used in the simulations.

To estimate the distribution for hypothesis testing we applied bootstrap resampling and estimated the empirical bootstrap distribution of the sample mean rewards produced by each ap-

[5]http://www.rollingstone.com/music/lists/500-greatest-albums-of-all-time-20120531

[6]As described earlier, the song features were derived from the entire song. In principle, it is possible that better results for DJ-MC can be produced if song features are derived from only the first 60 seconds to which users listened.

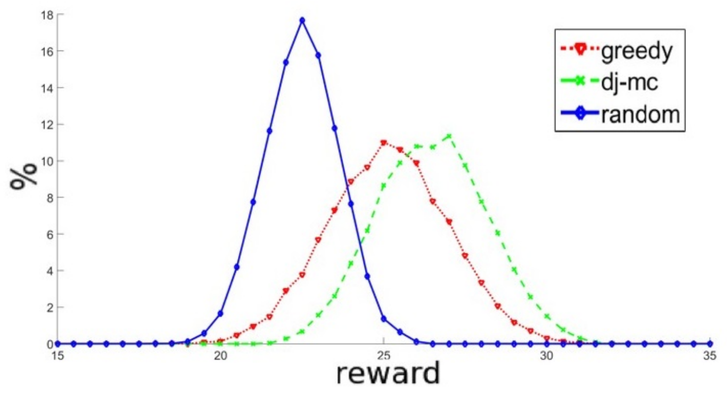


Figure 5. Histogram of Cumulative Song Rewards for RANDOM, GREEDY, and DJ-MC (Rewards after 25 steps; 25 songs played)

proach (for details on the bootstrapping procedure, see Efron 1979; Good 2006). Following the bootstrap method, the empirical bootstrap distribution of the aggregate reward is constructed for each approach by resampling the participants of the corresponding approach with replacement. Given a bootstrap estimate's accuracy increases with more bootstrap samples, the process was repeated $N = 10{,}000$ to produce the bootstrap distribution of the sample mean.

Figure 5 presents the histogram of human listener rewards achieved by the DJ-MC, GREEDY, and RANDOM approaches. When adapting to human listeners, DJ-MC produces a higher average reward than either GREEDY or RANDOM, and this improvement is statistically significant at $p << 0.05$. DJ-MC's performance demonstrates that DJ-MC's architecture indeed facilitates effective online adaptation to human listeners, whose preferences are likely to be both heterogeneous as well as arbitrarily complex relative to DJ-MC's internal representation of listeners' preferences. DJ-MC's superior rewards also suggest that human listeners' experiences are affected by the sequence in which songs are played. Recall that while GREEDY plays songs that it predicts will produce the highest reward, irrespective of the ensuing transitions, DJ-MC might trade off a high song reward to offer a more enjoyable sequence, or vice versa. We therefore examine whether DJ-MC achieves its advantage at the possible expense of lower song rewards, or whether it achieves higher song reward than does GREEDY, implying that DJ-MC's advantage cannot be attributed to its adaptation to listeners' sequential preferences.

We examined the transition rewards and song rewards obtained by each approach, separately. Recall that during the first 25 steps, both GREEDY and DJ-MC randomly explore the song space, and hence will achieve comparable song and transition rewards. Figures 6(a) and 6(b) confirm that the song and transition reward histograms for the two approaches yield comparable rewards at this phase. Figures 6(c) and 6(d) present the cumulative song and transition reward distribution during the learning and adaptation phase, when the approaches differ in the songs and sequences they choose to play to listeners. As shown in Figure 6(*c*), DJ-MC and GREEDY yield comparable song rewards; thus, the observed differences in their overall performance can be attributed exclusively to DJ-MC's modeling of listeners' transition preferences. Importantly, Figure 6(d) demonstrates that DJ-MC achieves higher transition reward than GREEDY, and this difference is statistically significant ($p << 0.01$).[7] These differences in transition rewards demonstrate that DJ-MC's learning and adaptation to human listeners' transition preferences yields more enjoyable sequences, and this advantage does not come at the expense of lower song rewards.

Our results with human listeners simultaneously show that the existence of patterns in transition preferences that DJ-MC captures and subsequently exploits to generate playlists implies that listeners indeed have preferences for transitions and that learning these patterns of preferences yields an overall better experience for listeners. Importantly, DJ-MC's design facilitates effective online learning and adaptation to human listeners' preferences, which are unknown, likely heterogeneous across listeners, and can be arbitrarily complex relative to DJ-MC's representation of listeners. As demonstrated in our simulation results as well, these benefits can be attributed directly to DJ-MC's simple representation of listeners that facilitates learning from limited online experiences.

[7]Testing whether the difference in mean reward is greater than is statistically significant.

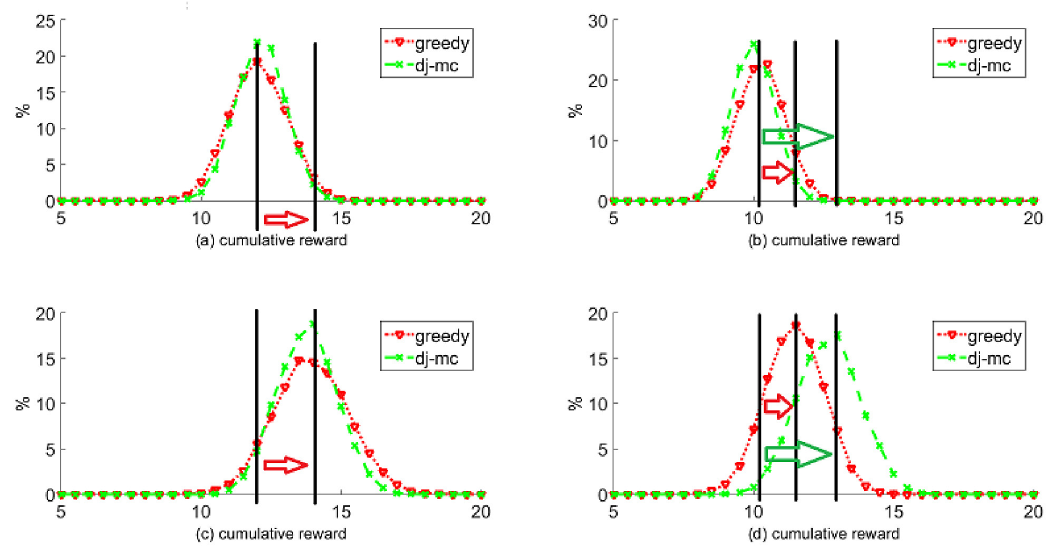


Figure 6. Histograms of Cumulative Rewards ((a) Histogram of cumulative *song* rewards during random exploration. (b) Histogram of cumulative *transition* rewards during random exploration. (c) Histogram of cumulative *song* rewards during learning and adaptation. (d) Histogram of cumulative *transition* rewards during learning and adaptation. Offsets representative of learning effects are marked across figures.)

Extensions to and Additional Analyses of DJ-MC

In Appendix A we first consider two extensions of DJ-MC that account for song costs (royalty fees) when producing a playlist. Another variant incorporates the listener's recent feedback explicitly in the information used by DJ-MC to infer the best next song to play. A third DJ-MC variant uses the listener's reward model to *deterministically* select the next song that yields the highest *immediate* reward. We find that the latter two variants do not benefit DJ-MC's performance and discuss the reasons for these results. In Appendix B we describe six additional studies that explore DJ-MC's performance in different settings. These include different planning horizons, different numbers of bins used to discretize song features, a different proportion, B, of top songs considered during planning, and a different number of songs to poll listeners during initialization in the simulation studies. We find that DJ-MC's performance is not very sensitive to these parameter settings.

Computational Complexity

DJ-MC's online performance is designed to provide viable recommendations in real time. Specifically, updating the listener model is linear in the number of feature song and feature transition bins, having a runtime complexity of $O(|F_s| \cdot nbins_s^2)$. The second online component, planning, is an anytime algorithm, which can be interrupted at any time and produce a valid song recommendation. Generating q random songs is done in constant time O(1), and evaluation of each sequence requires computing the reward, with complexity of $O(|K^2|F_s| \cdot nbins_s^2)$, for K song playlist. On a standard laptop with a 2.4 GHz Intel Core 17 processor and 8 GB RAM, DJ-MC ran 10,000 sequence rollouts in less than a second. In practice, a song may play for several minutes, during which time it is possible to generate and evaluate a significantly larger number of rollouts. Our evaluations demonstrate that DJ-MC's planning is both feasible in practice and that it yields better listening experiences for listeners.

Conclusions and Future Work

In this paper, we consider the general problem of playlist generation, we posit that *transition* preferences are integral to a listener's enjoyment from a playlist, and propose a framework for online learning and adaptation to sequential preferences within a listening session, so as to tailor the playlist to the listener's current context. Because learning is done from limited interactions with the listener, we propose that a simple model for representing the listener song and sequential preferences is advantageous to expedite generalization in our online setting; our data representation also employs factored representation of songs and sparse, binary representation of song features to facilitate fast generalizing. Experiments in simulation, seeded by real playlists as well as with human listeners, demonstrate that DJ-MC's design yields robust performance for listeners, whose preferences can be both heterogeneous and arbitrarily complex relative to DJ-MC's own representation of the listeners' preferences. In addition, DJ-MC yields better listening experiences to human listeners. Future work can also build on our contributions to consider other temporal aspects of preferences for which real-time adaption can be useful. In particular, DJ-MC's framework can be used to adaptively learn and recommend sequences of news items, videos, or television shows.

Future work can build on our contributions to improve transition preferences learning and playlist adaptation. In particular, it will be fruitful to explore alternative means to initiate the listener model. Active learning can be used early on to acquire feedback on particularly informative transitions for learning. Producing better sequences earlier may also benefit from improvements in planning. Specifically, the search space may be more efficiently explored by exploiting the structure of the song space, such as by learning song types, and then searching and planning a sequence of abstract song-types. Concrete songs can then be drawn from each song-type in the trajectory. Initialization may also be improved by leveraging prior knowledge (e.g., Pardoe et al. 2010), such as by initializing the listener model via learning from historical data the focal user's or even similar users' preferences across contexts.

Finally, reinforcement learning is a natural progression toward artificial intelligence techniques that learn and act on their own to achieve a goal exclusively by interacting with their environment. Such capabilities can benefit challenges arising in a variety of business environments. Financial markets and smart electricity markets are prime examples of environments where the fast-changing conditions can benefit from intelligent agents, acting on behalf of users to independently learn and adapt their strategies to benefit their users' goal (Peters et al. 2014). We hope that this research will inspire future research in IS that will identify and address new and important challenges in online, data-driven learning and decision making arising in a variety of dynamic business settings.

References

Adomavicius, G., and Kwon. Y. 2014. "Optimization-Based Approaches for Maximizing Aggregate Recommendation Diversity," *INFORMS Journal on Computing* (26:2), pp. 351-359.

Adomavicius, G., and Tuzhilin, A. 2005. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. Knowledge and Data Engineering," *IEEE Transactions on Knowledge and Data Engineering* (17:6), pp. 734-749.

Adomavicius, G., Tuzhilin, A., and Zheng, R. 2011. "REQUEST: A Query Language for Customizing Recommendations," *Information Systems Research* (22:1), pp. 99-117.

Adomavicius, G., and Zhang, J. 2012. "Stability of Recommendation Algorithms," *ACM Transactions on Information Systems* (30:4), Article 23.

Aizenberg, N., Koren, Y., and Somekh, O. 2012. "Build Your Own Music Recommender by Modeling Internet Radio Streams," in *Proceedings of the 21st International Conference on World Wide Web*, New York: ACM.

Berenzweig, A., Logan, B., Ellis, D. P., andWhitman, B. 2004. "A Large-Scale Evaluation of Acoustic and Subjective Music-Similarity Measures," *Computer Music Journal* (28:2), pp. 63-76.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. 2011. "The Million Song Dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference*, October 24-28, Miami, FL, pp. 591-596.

Berz, W. L. 1995. "Working Memory in Music: A Theoretical Model," *Music Perception* (12:3), pp. 353-364.

Chen, S., Moore, J. L., Turnbull, D., and Joachims, T. 2012. "Playlist Prediction via Metric Embedding," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York: ACM, pp. 714-722.

Chi, C. Y., Tsai, R. T., Lai, J., and Hsu, J. Y. 2010. "A Reinforcement Learning Approach to Emotion-Based Automatic Playlist Generation," *International Conference on Technologies and Applications of Artificial Intelligence*, IEEE, pp. 60-65.

Cliff., D. 2000. "Hang the DJ: Automatic Sequencing and Seamless Mixing of Dance-Music Tracks," Digital Media Systems Department, Hewlett-Packard Laboratories.

Davies, J. B. 1978. *The Psychology of Music*, London: Hutchinson.

Efron, B. 1979. "Bootstrap Methods: Another Look at the Jackknife," *The Annals of Statistics* (7), pp. 1-26.

Ghoshal, A., and Sarkar, S. 2014. "Association Rules for Recommendations with Multiple Items," *INFORMS Journal on Computing* (26:3), pp. 433-448.

Good, P. I. 2006. *Permutation, Parametric, and Bootstrap Tests of Hypotheses*, New York: Springer Science & Business Media.

Hastie, T., Tibshirani, R., and Friedman, J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.), Berlin: Springer-Verlag.

Harir, N., Mobasher, B., and Burke, R. 2012. "Context-Aware Music Recommendation Based on Latenttopic Sequential Patterns," in *Proceedings of the 6th ACM Conference on Recommender Systems*, New York: ACM, pp. 131-138.

Kahnx, B., Ratner, R., and Kahneman, D. 1997. "Patterns of Hedonic Consumption Over Time," *Marketing Letters* (8:1), pp. 85-96.

Kaji, K., Hirata, K., and Nagao, K. 2005. "A Music Recommendation System Based on Annotations About Listeners' Preferences and Situations," in *Proceedings of the 1st International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, IEEE.

King, J., and Imbrasaitė, V. 2015. "Generating Music Playlists with Hierarchical Clustering and Q-Learning," in *Proceedings of the European Conference on Information Retrieval: Advances in Information Retrieval*, A. Hanbury, G. Kazai, A. Raubner, and N. Fuhr (eds.), New York: Springer, pp. 315-326.

Liebman, E., Chor, B., and Stone, P. 2015. "Representative Selection in Nonmetric Datasets," *Applied Artificial Intelligence* (29:8), pp. 807-838.

Liebman, E., Saar-Tsechansky, M., and Stone, P. 2015. "DJ-MC: A Reinforcement-Learning Agent for Music Playlist Recommendation," in *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, E. Bordini and Y. Weiss (eds.), Istanbul, Turkey, pp. 591-599.

Maillet, F., Eck, D., Desjardins, G., and Lamere, P. 2009. "Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists," in *Proceedings of the 10th International Conference on Music Information Retrieval*, Kobe, Japan, pp. 345-350.

McFee, B., and Lanckriet, G. R. 2011. "The Natural Language of Playlists," in *Proceedings of the 12th International Conference on Music Information Retrieval*, Miami, FL, pp. 537-542.

Natarajan, N., Shin, D., and Dhillon, I. S. 2013. "Which App Will You Use Next? Collaborative Filtering with Interactional Context," in *Proceedings of the 7th ACM Conference on Recommender Systems*, New York: ACM, pp. 201-208.

Palmer, C. 2005. "Sequence Memory in Music Performance," *Current Directions in Psychological Science* (14:5), pp. 247-250.

Pardoe, D., Stone, P., Saar-Tsechansky, M., Keskin, T., and Tomak, K. 2010. "Data-Driven Auction Design and the Incorporation of Prior Knowledge," *INFORMS Journal on Computing* (22:3), pp. 353-370.

Prawesh, S., and Padmanabhan, B. 2014. "The 'Most Popular News' Recommender: Count Amplification and Manipulation Resistance," *Information Systems Research* (25:3), pp. 569-589.

Peters, M., Ketter, W., Saar-Tsechansky, M., and Collins, J. 2013. "A Reinforcement Learning Approach to Autonomous Decision-Making in Smart Electricity Markets," *Machine Learning* (92:1), pp. 5-39.

Platt, J. C. 2003. "Fast Embedding of Sparse Similarity Graphs," in *Proceedings of the Advances in Neural Information Processing Systems Conference*, Vancouver, BC, Canada.

Shivaswamy, P. K., and Thorsten, J. 2011. "Online Learning with Preference Feedback," paper presented at the NIPS Workshop on Choice Models and Preference Learning.

Saar-Tsechansky, M., Melville, P., and Provost, F. 2009. "Active Feature-Value Acquisition," *Management Science* (55:4), pp. 664-684.

Saar-Tsechansky, M., and Provost, F. 2007. "Decision-Centric Active Learning of Binary-Outcome Models," *Information Systems Research* (18:1), pp. 4-22.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning*, Cambridge, MA: MIT Press.

Tan, S.-L., Pfordresher, P., and Harré, R. 2010. *Psychology of Music: From Sound to Significance*, Hove, England: Psychology Press.

Urieli, D., and Stone, P. 2013. "A Learning Agent for Heat-Pump Thermostat Control," in *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, St. Paul, MN.

Wang, X., Wang, Y., Hsu, D., and Wang, Y. 2013. "Exploration in Interactive Personalized Music Recommendation: A Reinforcement Learning Approach," *ACM Transactions on Multimedia Computing, Communications and Applications* (2:3), Article 1.

Weston, J., Bengio, S., and Hamel, P. 2011. "Multi-Tasking with Joint Semantic Spaces for Large-Scale Music Annotation and Retrieval," *Journal of New Music Research* (40:4), pp. 337-348.

Zheleva, E., Guiver, J., Mendes Rodrigues, E., and Milić-Frayling, N. 2010. "Statistical Models of Music-Listening Sessions in Social Media," in *Proceedings of the 19th International Conference on World Wide Web*, New York: ACM, pp. 1019-1028.

About the Authors

Elad Liebman completed his Ph.D. at the Department of Computer Science, UT Austin, advised by Peter Stone. His research focuses on sequential decision-making problems related to content recommendation, and multiagent systems, with a special emphasis on music recommendation, musical preference learning, and preference elicitation in cultural settings in general. He is also interested in modeling the effects of musical stimuli on human decision making, and agents learning to interact with other agents (human or artificial) in social settings. In his years in the field he has worked on a wide range of problems, spanning the gamut from bioinformatics and telecommunications to finance and robotics. He is currently a senior data scientist at SparkCognition Inc.

Maytal Saar-Tsechansky is an associate professor of Information, Risk and Operations Management at the McCombs School of Business, University of Texas at Austin. Her research focuses on the development of novel machine learning and artificial intelligence methods with the goal of improving decision making and to benefit people, organizations, and society. Her research integrates business, machine learning, and artificial intelligence, and she has addressed challenges in a variety of domains, including health care, smart electricity grid, fraud detection, finance, and emerging forms of work, such as online labor markets. Maytal received her Ph.D. from New York University's Stern School of Business. Her research has been published in business and computer science journals, including *Journal of Finance, Management Science, Information Systems Research, Journal of Machine Learning Research*, and *Machine Learning Journal*. Her research has been supported by both government and industry, including the National Science Foundation, SAP, and the Israeli Science Ministry. Maytal co-founded Sweetch.com, a start-up company focused on predicting the onset and prevention of chronic diseases.

Peter Stone is the David Bruton, Jr. Centennial Professor and Associate Chair of Computer Science, as well as Chair of the Robotics

Portfolio Program, at the University of Texas at Austin. In 2013 he was awarded the University of Texas System Regents' Outstanding Teaching Award and in 2014 he was inducted into the UT Austin Academy of Distinguished Teachers, earning him the title of University Distinguished Teaching Professor. Peter's research interests in Artificial Intelligence include machine learning (especially reinforcement learning), multiagent systems, robotics, and e-commerce. He received his Ph.D. in Computer Science in 1998 from Carnegie Mellon University. From 1999 to 2002 he was a senior technical staff member in the Artificial Intelligence Principles Research Department at AT&T Labs–Research. He is an Alfred P. Sloan Research Fellow, Guggenheim Fellow, AAAI Fellow, IEEE Fellow, AAAS Fellow, Fulbright Scholar, and 2004 ONR Young Investigator. In 2003, he won an NSF CAREER award for his proposed long term research on learning agents in dynamic, collaborative, and adversarial multiagent environments, in 2007 he received the prestigious IJCAI Computers and Thought Award, given biannually to the top AI researcher under the age of 35, and in 2016 he was awarded the ACM/SIGAI Autonomous Agents Research Award. Peter co-founded Cogitai, Inc., a start-up company focused on continual learning, in 2015, and currently serves as President and COO.

THE RIGHT MUSIC AT THE RIGHT TIME: ADAPTIVE PERSONALIZED PLAYLISTS BASED ON SEQUENCE MODELING

Elad Liebman
SparkCognition, Inc., 4030 West Braker Lane #500, Austin, TX 78759 U.S.A. {eladlieb@gmail.com}

Maytal Saar-Tsechansky
McCombs School of Business, The University of Texas at Austin, 2110 Speedway, Stop B6500, Austin, TX 78712-1277 U.S.A. {maytal@mail.utexas.edu}

Peter Stone
Department of Computer Science, The University of Texas at Austin, 2317 Speedway, Stop D9500, Austin, TX 78712-1277 U.S.A. {pstone@cs.utexas.edu}

Appendix A

Extending DJ-MC to Incorporate Song Costs or Royalty Fees

In this appendix, we explore extensions of DJ-MC to accommodate song costs, as well as to evaluate the performance of several variants of DJ-MC.

An issue rarely explored in music recommendation is how to incorporate song costs in the content recommendation process. Not much is known about the particular royalty cost of playing each song in existing streaming services. Studies indicate that streaming services often negotiate with royalty agreements agencies,[1] and the royalties are typically determined in bulk, involving a large number of artists and songs, and are not assigned to each song individually. For example, Spotify, a leading streaming service, has recently revealed that it does not assign a different royalty fee to each song, but rather the payment to owners (labels, publishers, distributors, etc.) is determined *ex post* by the proportion of streams made relative to Spotify's total revenue from the corresponding period.[2] Nevertheless, given that the music industry is experiencing a transformation, it may also be that future models incorporate differential, individual song fees. Hence, in this section we propose how to incorporate individual song costs in the DJ-MC framework.

In principle, given a (mathematical) mapping between listener enjoyment and monetary value that the playlist service can expect, $Monetary(R)$, DJ-MC can incorporate song fees to optimize monetary returns directly and seamlessly. Once such mappings are established, song rewards R_s and song costs C_s inhabit the same (monetary) space, and DJ-MC can then operate as before, using a cost-adjusted reward model $R^* = Monetary(R) - C$. However, in the absence of an established mapping, one can consider alternative approaches with goals other than maximizing profit for streaming services to aim for.

[1]http://journals.law.stanford.edu/sites/default/files/stanford-technology-law-review/online/licensinginshadow.pdf

[2]https://www.spotifyartists.com/spotify-explained/#how-we-pay-royalties-overview

The first approach we consider selects the next song such that it aims to maximize the *ratio* between expected listener reward and song cost. Thus, in Algorithm 4 we now aim to optimize the ratio between the reward and the cost:

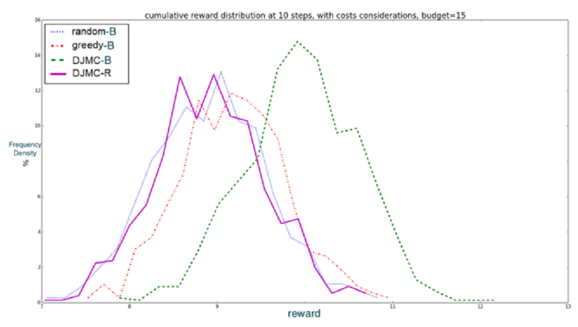
$$R_s\left(song_i\right)/C_s\left(song_i\right)+\Sigma_{i=2}^{q}\left(R_t\left(\left(song_1,\ldots,song_{i-1}\right)song_i\right)+R_s\left(song_i\right)\right)\Big/C_s\left(song_i\right)$$

This approach, henceforth referred to as DJ-MC-R (DJ-MC-RATIO), aims to promote the selection of songs that yield greater marginal enjoyment per unit cost. While DJ-MC-R may seem intuitive for handling the tradeoff between listener reward and cost, note that, because listener enjoyment is no longer the only criterion, to yield a high ratio, DJ-MC-R may also select sequences composed of songs (and transitions) that incur very low costs, even if the listener may not particularly enjoy them.

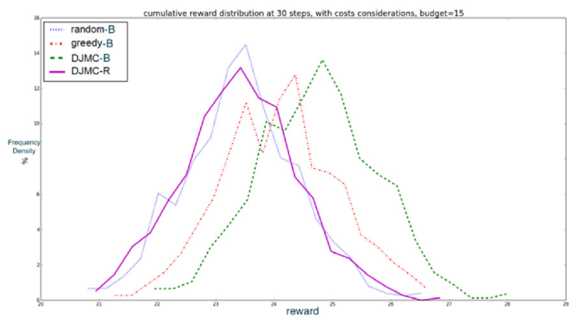
A second alternative we propose avoids a direct tradeoff between enjoyment and costs altogether. Specifically, it aims to produce sequences that yield the highest possible rewards *within a budget*, where a budget corresponds to the maximum cumulative fees one is willing to incur for a single listener over a given period of time. Henceforth, we refer to this variant as DJ-MC-B (DJ-MC-BUDGET), and we adapt DJ-MC to this framework by imposing a *budget constraint* on the exploration and planning process described in Algorithm 4. Specifically, in Algorithm 3, DJ-MC-B selects each song in the trajectory in the same way, but subject to the condition that $Cost(trajectory \cup \{song\}) \leq budget$. We similarly adapt the RANDOM and GREEDY baselines to create playlists that do not exceed the allocated budget. Specifically, the RANDOM baseline is adapted to accept only RANDOM sequences complying with the budget constraint. Similarly, GREEDY selects songs as before; however, if the next best song leads to a playlist that exceeds the budget, the song is skipped and the next highest-reward song is attempted. The budget-aware variants of GREEDY and RANDOM are henceforth referred to as GREEDY-B and RANDOM-B, respectively.

In the experiments reported below, song fees per stream for a given song are drawn uniformly from [0,1]. In addition, we identified that a budget of 15 imposes a meaningful constraint on the songs that can be played over the course of a session, such that methods that do not account for the cost of songs exhaust the budget in a significant proportion of the sessions. We therefore used a budget of 15 to explore the listener rewards produced by each approach. Finally, we examine separately the rewards and the remaining budget for each approach to draw conclusions on the tradeoffs offered by each.

Figures A1(a) and A1(b) show the listener reward distribution produced by each approach after 10 and then 30 songs, given a budget of 15. Together, Figures A1(a) and A1(b) show that both DJ-MC-B and DJ-MC-R yield higher remaining budgets in expectation as compared to the RANDOM and GREEDY benchmarks. This can be attributed to the fact that both DJ-MC-B and DJ-MC-R plan ahead, given they consider sequences, and are therefore less likely to exhaust their budget while generating the playlist. Because DJ-MC-R economizes on costs directly, it yields the highest remaining budget (uses a smaller proportion of its budget) as compared to DJ-MC-B. As expected, because DJ-MC-R selects songs for which the ratio between expected reward and cost is high, this also leads to the selection of songs that are inexpensive even if the listener may not particularly enjoy them. Consequently, the cumulative listener reward produced by DJ-MC-R is lower as compared to DJ-MC-B and GREEDY-B, whose primary criterion is to select songs that are likely to maximize listener reward.



(a) Listener Reward Distribution After 10 Steps

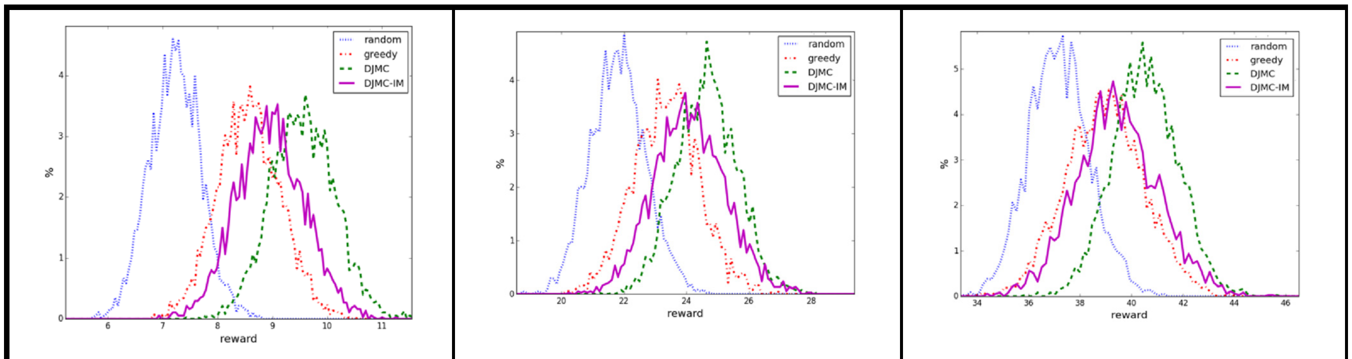


(b) Listener Reward Distribution After 30 Steps

Figure A1. for the Simple Model (Uncorrelated LVs)

DJ-MC *with Only Immediate Reward*

Recall, because DJ-MC simultaneously *learns* and *acts*, its planning aims to manage a tradeoff between two goals: explore the listener's preferences, and exploit its current knowledge of the listener's preferences, so as to select a song that will be likely followed by an enjoyable sequence. To explore, DJ-MC simulates RANDOM sequences of songs. To assess the value of each simulated sequence, DJ-MC uses the reward model to (1) compute the *immediate* reward from playing the first song in the sequence, and (2) to assess how the next song may affect the enjoyment from the trajectory of future songs. Figure A2 shows results for a variant of DJ-MC, denoted DJ-MC-IMMEDIATE (DJ-MC-IM), in which DJ-MC uses its listener reward model of song and transition preferences to deterministically select the single song that yields the highest *immediate* reward. Hence, DJ-MC-im does not explore the listener's preferences to improve its learning of these preferences, and it does not consider how the choice of the next song may affect the listener's enjoyment from future songs.



Performance after 10 steps | Performance after 30 steps | Performance after 50 steps

Figure A2. Evaluation of DJ-MC-IM

As shown, consistent with our prior results for DJ-MC, DJ-MC- im's learning and accounting for transition preferences when selecting the next song remains advantageous over the GREEDY approach, which selects the next song based on the *song* reward exclusively. The comparison between DJ-MC and DJ-MC-IM also sheds light on the benefits of the standard DJ-MC's exploration of the listener's preferences via the RANDOM rollouts, as well as DJ-MC's considerations of not only the immediate song and transition reward (namely, the transition from songs played thus far onto the next song), but how the choice of the next song may affect the enjoyment from future songs in the playlist. Specifically, via exploration, DJ-MC aims to select songs that improve the listener reward model and the selection of future songs. By evaluating the rewards from future trajectories of songs, the standard DJ-MC is also enabling the possibility of not selecting a song with the best immediate song and transition reward, in order to select songs that will yield a more enjoyable sequence. As shown in Figure A2, DJ-MC yields better rewards already after 10 steps relative to the myopic variant, DJ-MC-IM.

Notes on Incorporating User Feedback in the State Space

One may consider representing user feedback in the state explicitly so as to enable DJ-MC to learn different models of behavior in the context of different listener feedback. This strategy can improve DJ-MC's performance by, for example, being more conservative and avoiding explorations of the listener's preferences when the listener is indicating dissatisfaction. However, feedback is already being reflected implicitly in DJ-MC's recommendations. For example, as reflected in Algorithm 3, the update weight extracted from the reward at step k, is given by $\log(v_k/\bar{v}_k)$, where $\bar{v}$ is the running average of reward, reflecting recently observed feedback. Because after a sequence of negative rewards, the average, $\bar{v}$, decreases, a positive reward translates into a significant increase in the reward computed for songs the listener enjoys—more so than it would have been in a less negative context. Consequently, after a "bad run," the model is quicker to adapt to any evidence of more favorable song preferences and, similarly, after a "good run," it is slower to do so, requiring more favorable feedback in order to make positive adjustments. Therefore, this property of DJ-MC results in a higher likelihood in such a context to recommend songs the listener is likely to enjoy and, similarly, rendering DJ-MC less likely to explore a listener's taste by recommending less enjoyable songs.

Appendix B

Additional Analyses

In this appendix, we report additional studies we have conducted to assess its performance under different parameter settings.

Planning Horizons

Our DJ-MC implementation uses a planning horizon, q, of 10 songs, while in practice, the true playlist horizon at any time can be arbitrarily longer. We therefore compared DJ-MC's performance with a planning horizon of 10 songs, to its performance with planning horizons of 5 or 30 songs.

As shown in Figure B1, we find that planning horizons of 5, 10 (as before), and 30 songs yield comparable results. Note that all the variants evaluated here perform stochastic exploration and an assessment of the likely effect of the next song on enjoyment from future songs in the playlist. Our earlier results show that not performing explorations, and myopically selecting the next song that yields the best immediate song and transition reward, yields worse rewards.

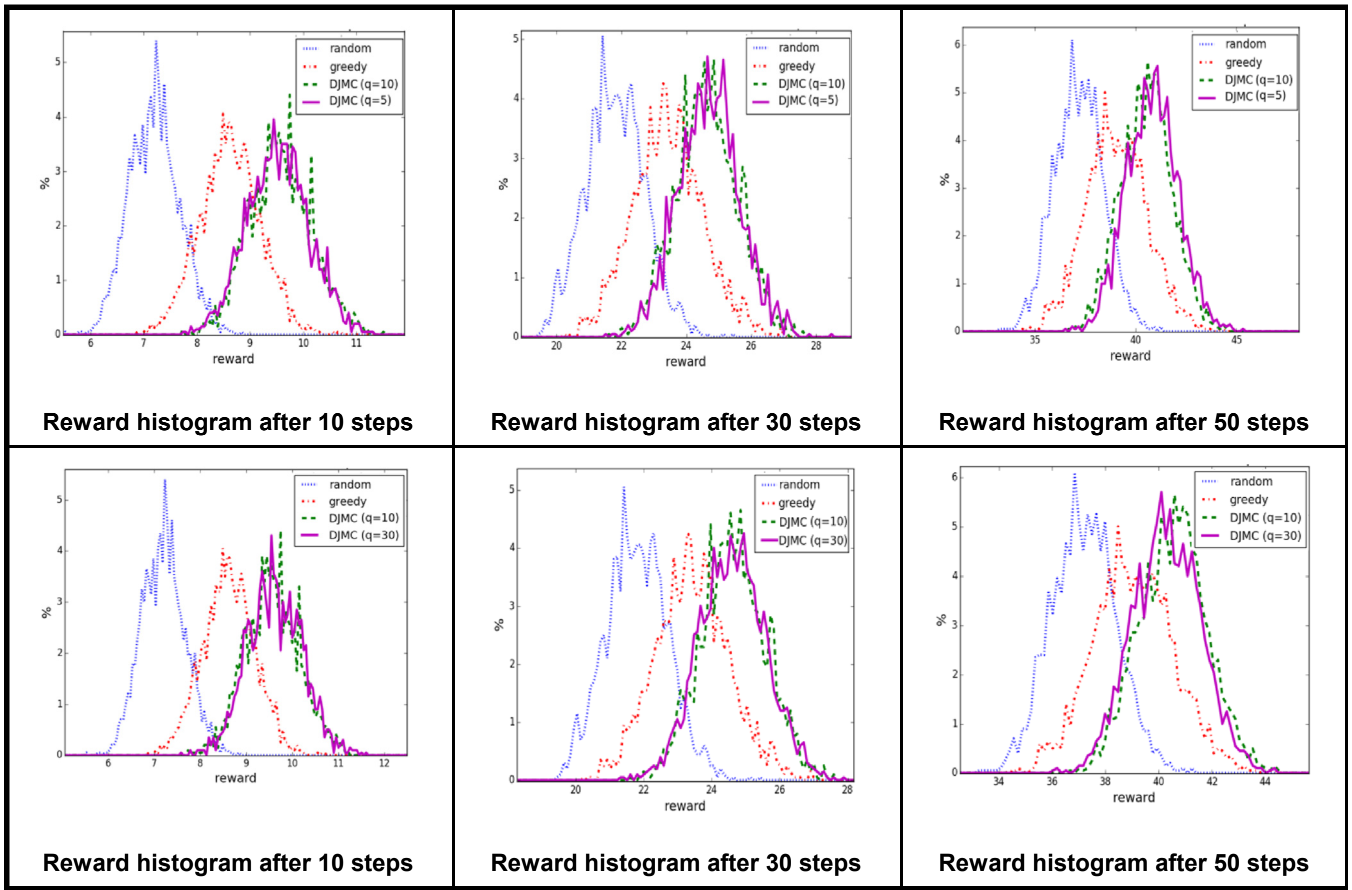
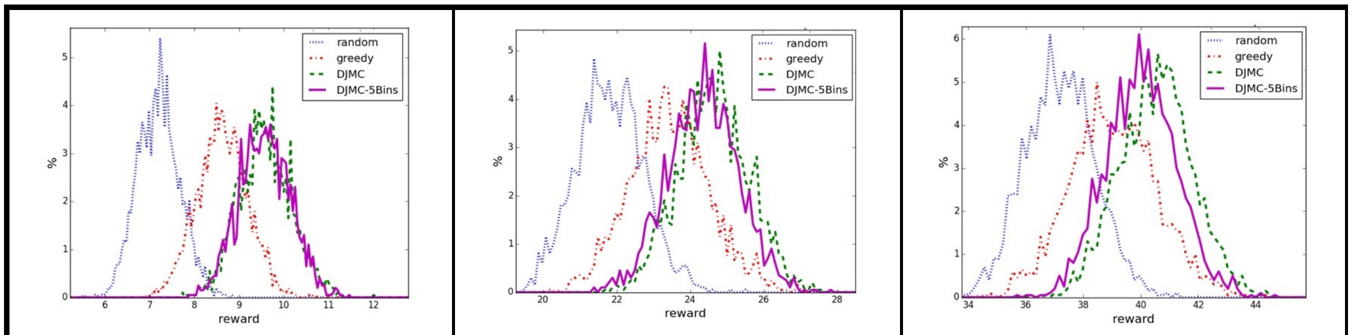


Figure B1. DJ-MC's Performance Using Different Planning Horizons During Planning (First row: Comparison of DJ-MC with planning horizons (q) of 10 and 5 songs. Second row: comparison of DJ-MC with planning horizons (q) of 10 and 30 songs.)

Using Coarser Binning in the Song Representation

When considering the complexity of the listener reward model, we discuss the tradeoff between the model's bias and variance given the amount of *experiential* data, namely the amount of actual experiences with the listener. Fitting a reward model in our setting is constrained by the number of actual experiences with the listener during a listening session, namely between 10 and 50 songs. In principle, given unlimited training experiences, having a more complex (higher variance) reward model allows us to fit the listener's preferences better; however, if the number of training experiences is insufficient, this flexibility becomes a liability, and the reward model can over-fit the limited training experiences. Our study of a feature-dependent reward model, with the ability to model how a very large number of interactions between all possible pairs of song features maps onto preferences, shows that, given the number of experiences with the listener in our setting, such a model over-fits these training experiences. Thus, we find in our experiments that using such a listener reward model undermines DJ-MC's performance, even when the model reflects the true patterns underlying listeners' preferences.

The song representation adds another dimension by which we can control the reward model's complexity. The first element is the binning itself, so that we represent songs by the corresponding percentile bin value for each feature, rather than the corresponding feature's precise value. In addition, the choice of number of bins can vary as well. As shown in Figure B2, a comparison between DJ-MC with 5 bins and the standard DJ-MC with 10 bins, shows that the latter does not over-fit relative to the former. Thus, we find that having fewer bins does not introduce quite as dramatic a change to the variance of the model.

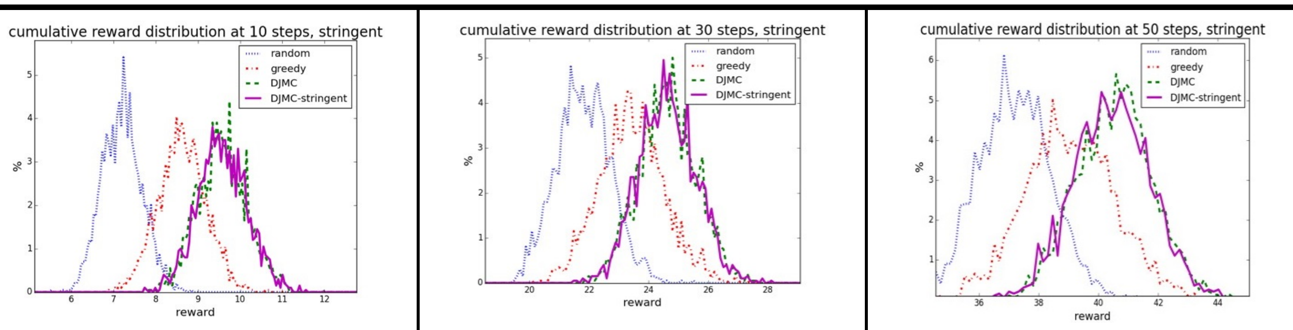


Reward histogram after 10 steps **Reward histogram after 30 steps** **Reward histogram after 50 steps**

Figure B2. DJ-MC's Performance with Coarser Feature Value Binning

Using Smaller Subset, B, of Top Songs During Planning

During planning, DJ-MC produces rollouts, namely RANDOM sequences from the top B = 50% of the listener's most favorable songs (as estimated by DJ-MC) so as to assess the likely effect of the first song in the sequence on enjoyment from subsequent songs. We explored the sensitivity of our results to using a smaller subset of the top 25% most enjoyable songs in M. This reduction in the song space primarily increases the likelihood of selecting an enjoyable song; however, it might not improve the likelihood of identifying an enjoyable sequence and corresponding transitions. As show in Figure B3, our results suggest that reducing the set to 25% yields comparable performance for DJ-MC.

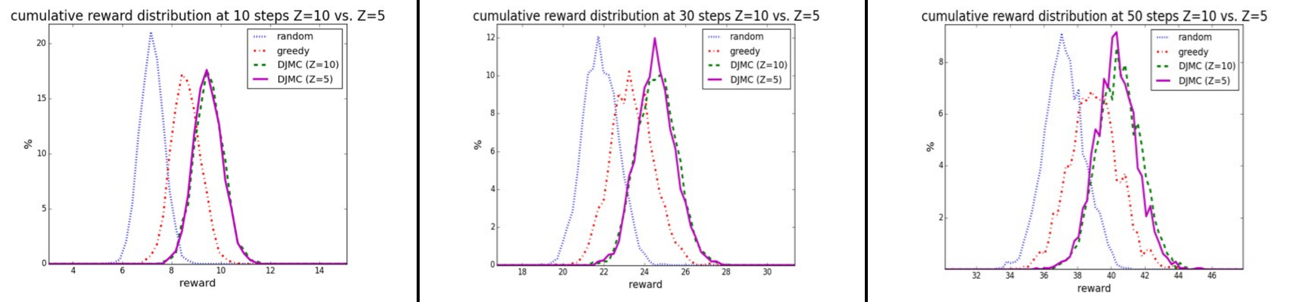


Reward histogram after 10 steps | Reward histogram after 30 steps | Reward histogram after 50 steps

Figure B3. DJ-MC's Performance Using a Smaller Subset of Top Songs During Planning

Polling Listeners on Fewer Songs and Transitions During Initialization in Simulation Studies

In the simulation studies, a listener's reward model is first initialized with uniform weights so as to reflect that all songs and transitions are equally desirable. The initialization proceeds to poll the listener for songs she prefers and then asking the listener select a short sequence from which transition preferences are initialized. In the experiments we report in the paper, the (simulated) listener is polled for 10 songs and transitions. Figure B4 shows that when the number of songs and transitions the listener is polled for is five, the results are only slightly worse initially, but not meaningfully different from the results we report in the body of the paper. In the conclusions section, we discuss future work to explore the use of prior knowledge during initialization, so as to allow production of better playlists earlier.



Reward histogram after 10 steps | Reward histogram after 30 steps | Reward histogram after 50 steps

Figure B4. DJ-MC's Performance When Listeners are Polled for 5 and 10 Songs and Transitions During Initialization