"Reinventing the Wheel": A Novel Approach to Music Player Interfaces

Tim Pohle, Peter Knees, Markus Schedl, Elias Pampalk, and Gerhard Widmer

***Abstract*—We present a novel interface to (portable) music players that benefits from intelligently structured collections of audio files. For structuring, we calculate similarities between every pair of songs and model a travelling salesman problem (TSP) that is solved to obtain a playlist (i.e., the track ordering during playback) where the average distance between consecutive pieces of music is minimal according to the similarity measure. The similarities are determined using both audio signal analysis of the music tracks and web-based artist profile comparison. Indeed, we will show how to enhance the quality of the well-established methods based on audio signal processing with features derived from web pages of music artists. Using a TSP allows for creating circular playlists that can be easily browsed with a wheel as input device. We investigate the usefulness of four different TSP algorithms for this purpose. For evaluating the quality of the generated playlists, we apply a number of quality measures to two real-world music collections. It turns out that the proposed combination of audio and text-based similarity yields better results than the initial approach based on audio data only. We implemented an audio player as Java applet to demonstrate the benefits of our approach. Furthermore, we present the results of a small user study conducted to evaluate the quality of the generated playlists.**

***Index Terms*— Feature extraction, music, music playlist generation, portable media players, user interfaces.**

I. Introduction

OVER the past few years, electronic music distribution (regardless of whether commercial or personal) has led many users to accumulate vast collections of digital audio files. These large numbers of audio tracks in private repositories make it virtually impossible for users to keep track of every piece. Thus, sophisticated methods for organizing large music repositories become more and more important. Considering the overwhelming economic success of high-capacity mobile music players, such as Apple's "iPod", the need for intelligent methods for structuring and exploring music collections in this domain becomes apparent. Since mobile devices are intended to be used en-route, interface designers have to consider that users are seldom capable of paying much attention to the handling of their players. For this reason, simple and intuitive—nevertheless powerful—user interfaces will be a vital part of future music players.

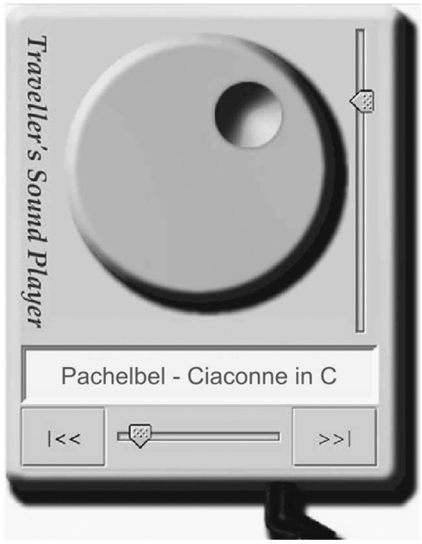


Fig. 1. Screenshot of our Java applet "Traveller's Sound Player".

For example, imagine the following situation: a passionate jogger always does sports with music, but is fed up with listening to the same music over and over again. Therefore, she purchases a mobile MP3-player with a large hard disk. Unfortunately, the songs are not structured intelligently such that similarly sounding pieces are arranged adjacently, but in a traditional artist-album-track hierarchy. Thus, the jogger has to stop and manually change the order in which the music pieces should be played (the *playlist*).

Using our novel user interface, she could choose music which fits her momentary mood, without delaying the workout, by browsing through the collection with a circular wheel. Such a user interface could look like the one in Fig. 1. After playing the track our potential user chose as a starting point, the player automatically keeps on playing similar tracks. At the end of her route, the user may simply turn the wheel to another music style on the fly, e.g., to some relaxing chill-out songs for walking home.

We present an approach to automatically organizing a music collection into a large circular playlist by applying a Travelling Salesman Algorithm on the calculated music similarity. If such an algorithm succeeds in finding the best tour, the generated playlist satisfies the constraint that consecutive tracks are

Manuscript received March 27, 2006; revised September 18, 2006. This work was supported by the Austrian Fonds zur Förderung der Wissenschaftlichen Forschung (FWF) under Project L112-N04 and by the Vienna Science and Technology Fund (WWTF), project CIO10 "Interfaces to Music". The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry for Education, Science, and Culture and by the Austrian Federal Ministry for Transport, Innovation, and Technology. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jie Yang.

T. Pohle, P. Knees, M. Schedl, and G. Widmer are with the Department of Computational Perception, Johannes Kepler University, Linz 2020, Austria (e-mail: tim.pohle@jku.at; peter.knees@jku.at; markus.schedl@jku.at; gerhard.widmer@jku.at).

E. Pampalk is with the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan (e-mail: elias.pampalk@gmail.com).

Digital Object Identifier 10.1109/TMM.2006.887991

maximally similar on average, i.e., the average distance between consecutive tracks is minimal. The whole playlist—and thus the whole collection—is easily accessible with only one circular controller which we call the "wheel".

Our approach is based on the calculation of acoustical similarity between music tracks and on retrieving information about artists from the web. This paper contributes to the current state-of-the-art by presenting a new way to combine these distinct sources of information for playlist generation.

For evaluation, we used two audio collections with different characteristics. It turns out that the proposed combination of audio- and text-based similarity not only reduces the number of necessary calculations considerably but also yields better results, in terms of musical quality, than the initial approach based on audio data only, cf. [1]. Additionally, we conducted a small user study that further confirms the quality of the resulting playlists.

II. Related Work

Quite some work has already been done on playlist generation as well as on user interfaces to music collections. [2] treat the problem of playlist generation as a network flow problem. Given a song collection, where each piece is labeled with a number of boolean attributes representing arbitrary aspects of the music, one start track and one end track, the algorithm finds a path (of user-defined length) through the network satisfying user-defined constraints. The proposed algorithm is an integer linear program, thus it is NP-hard.

In [3], a more efficient approach for handling various types of metadata is presented. According to user-defined constraints, the metadata of each track is transformed into a cost function. The playlist is constructed by iteratively optimizing an initial randomly chosen playlist with regard to the cost function.

In [4], it is not assumed that the tracks are already labeled. The playlist generation algorithm is rather based on a music similarity function (cf. [5]) which can be computed automatically.[1] Several approaches are evaluated for producing a playlist of given length for a given start track.

Regarding user interfaces that support browsing through music collections, there exist quite different approaches. [6] presents an approach where users can find music by specifying properties (e.g., tempo or spectral centroid). In [7], self-organizing maps (SOM) [8] are used to cluster similar pieces of music. The SOMs are visualized by means of smoothed data histograms, cf. [9]. One of the user interfaces also provides hierarchical structuring of music collections.

In [10], a highly interactive user interface that facilitates music exploration and playlist generation is presented. The user can grab pieces of music from similarity-based flows of tracks to create playlists.

III. Method

Comparing pieces of music is regarded as being a difficult task as musical similarity is an ill-defined concept. There are many different aspects of music similarity, whose actual impacts vary between individuals and depending on the situation. Although often used as a dimension for structuring music collections, musical genre lacks an objective and universal definition. Even though it may capture some aspects, intelligent structuring techniques should not solely rely on genre information.

Algorithms for capturing aspects of musical similarity—e.g., timbre, instrumentation, or socio-cultural background of composers and performers—have been developed. The outcome of such an algorithm can be converted into a low-dimensional space by applying dimensionality reduction techniques such as the principal component analysis (PCA) or self-organizing maps (SOMs), e.g., [11]. The low-dimensional representation can be used for playlist generation and browsing music collections.

In the approach we present here, we generate one large playlist consisting of all tracks from the collection by modeling a travelling salesman problem (TSP) on the musical distances obtained from an audio-based similarity function. We further enhance it by including web-derived features, which also incorporates "cultural" aspects since the web provides knowledge and opinions of a large number of people. The basic goal is to maximize the average similarity between consecutive tracks in the playlist, and thus, to obtain playlists containing large sections of consistent music. The playlist resulting from our algorithm can be interpreted as a projection of the collection onto one dimension. It is arranged around a circular wheel to make it easily accessible.

A. Audio-Based Similarity

For computing the perceived acoustical similarity of music, a great variety of algorithms has been proposed (for an overview of some of these, see e.g., [12]). In this work, we decided to follow a well-established algorithmic procedure, which we have shown in previous experiments to outperform many other approaches based solely on audio signal analysis [12]: For each audio track, mel frequency cepstral coefficients (MFCCs), e.g., [13], [14], are computed on short-time audio segments (called *frames*) to describe the spectral envelope of each frame. The n^{th} MFCC c_n is computed via the inverse Fourier transform of the log spectrum, with the spectrum S being represented on the mel scale [15]

$$c_n = \frac{1}{2\pi} \times \int_{\omega=-\pi}^{\omega=+\pi} \log\left(S(e^{j\omega})\right) \cdot e^{j\omega \cdot n} d\omega. \quad (1)$$

By discarding the higher-order MFCCs, it is possible to retain only a rather coarse description of the frame's envelope. This is beneficial for the ability to compare different frames, but possibly at the cost of discarding musically meaningful information.

Ignoring the temporal order of frames, each song is then represented as a Gaussian mixture model (GMM), e.g., [16], of the distribution of MFCCs. The similarity of two pieces is defined as the inverse of the distance between their GMMs, which again is computed by determining the likelihood that a number of points generated randomly with the distribution of one song's GMM would have been produced by the other song's GMM, and vice versa. Each of these steps is described in detail in the relevant

[1] In our experiments, we used a similar function.

literature (e.g., [5], [15], [17]). In our implementation, we used parameters consistent with those proposed in [17]. The wave files were downsampled to 22 kHz, on each frame we calculated 19 MFCCs, and the number of clusters per GMM was 30.

B. Web-Based Similarity

Perception of music is a highly subjective phenomenon that is also influenced by information other than the pure audio signal. Indeed, cultural, social, historical, and contextual aspects should be taken into account when trying to model human music perception. An easily accessible source for this kind of information is the World Wide Web since it incorporates many people's knowledge and opinions. Thus, to complement the audio-based similarity measure with cultural knowledge, we apply web-information retrieval techniques. To this end, we use the names of the artists contained in the collection. For each artist, we search the web with Google. The query string consists of the artist's name as an exact phrase extended by the keyword *music*. We retrieve 50 of the top-ranked web pages for each query, remove all HTML tags, and use common English stop word lists to remove frequent terms. For computational efficiency, we also remove all terms that do not occur on at least c pages over all artists. We choose the threshold c such that about 10 000 terms remain.

For each artist a and each term t appearing in the retrieved pages, we count the number of occurrences tf_{ta} (term frequency) of term t in documents related to a. Furthermore, we count df_t the number of pages the term occurred in (document frequency). These are combined using the term frequency $\times$ inverse document frequency $(tf \times idf)$ function [18]. The term weight per artist is computed as

$$w_{ta} = \begin{cases} (1 + \log_2 tf_{ta}) \log_2 \frac{N}{df_t}, & \text{if } tf_{ta} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where N is the total number of pages retrieved.

As a result, each artist is described by a vector of term weights. The weights are normalized such that the length of the vector equals 1 (cosine normalization). This removes the influence that the length of the retrieved web pages would otherwise have. Using this representation, similarities between artists can be derived, for example, by calculating Euclidean distances. However, combining this distance measure with the audio approach would not be practicable since only tracks by the same artist could be assigned the minimum possible distance. This would inhibit transitions between songs performed by different artists. Thus, we prefer a dichotomic way of combining the two similarity sources, which is elaborated upon next.

C. Combining Both Approaches

After having determined audio similarities between individual audio tracks and web-based artist profiles, we join both sources to create one distance matrix that serves as input to a TSP algorithm. In general, we adapt the audio-based track distances by adding penalties to the transitions between songs by dissimilar artists. More precisely, we redefine artist similarity using a SOM [8] that is trained on the set of web-based term weight vectors. We define two artists to be similar if they are mapped to the same unit or to adjacent units of the SOM. The 4 $\times$ 4 SOM resulting from our evaluation collection (cf. Section V) is depicted in Fig. 2. The size of the SOM was chosen arbitrarily; newer experiments have actually shown that 6 $\times$ 6 might be better for this collection. The penalty is a large constant value[2] that is added to the audio-based distance matrix for all songs of dissimilar artists in the sense explained above.

FOLK (80%) RAP (33%) PUNK (17%) JAZZ (11%)	ELECTRO (22%) FOLK (20%)	ACID JAZZ (25%) ELECTRO (6%)	PUNK (67%) METAL (50%) RAP (25%) ELECTRO (6%)
ITALIAN (60%) RAP (8%)	ITALIAN (20%)	REGGAE (33%) CELTIC (20%)	METAL (40%) PUNK (17%) ELECTRO (6%)
ITALIAN (20%) RAP (8%)		REGGAE (67%)	RAP (17%) ELECTRO (6%)
BLUES BOSSA NOVA JAZZ (89%) CELTIC (60%) A CAPELLA (50%)	A CAPELLA (50%) ACID JAZZ (25%) CELTIC (20%) METAL (10%) RAP (8%)	ACID JAZZ (25%) ELECTRO (3%)	ELECTRO (50%) ACID JAZZ (25%)

Fig. 2. The 4 $\times$ 4 SOM trained on the web-based features. For reasons of lucidity, genres instead of individual artists are depicted. The values in brackets represent the percentage of artists from the corresponding genre which are mapped to the respective map unit. If no value is given, the unit contains all artists from the respective genre.

D. Using the TSP for Playlist Creation

The task of generating a playlist based on the above similarity matrix is mapped to the TSP, which is a classical problem in computer science. In its basic form, it is formulated as follows: A salesman needs to visit n cities, each of them once. The distances (or costs) to travel between the cities are known. The problem is to find the optimal (i.e., shortest or cheapest) route to visit all cities exactly once, and return to the originating city.

In our setup, the cities (i.e., nodes of a graph) correspond to the tracks in the collection, and the distances (edges) are determined by the similarities between the tracks. Finding the optimal route means producing a circular playlist that contains all tracks of the collection, and in which the sum of the similarities along the path is maximized.

IV. TSP Algorithms

In this section, algorithmic aspects of the TSP are discussed. First, we give some general remarks, then we mention which aspects should be regarded in the specific case of similarity data, and finally we give a short description of the algorithms we evaluated.

[2]We used one half of the maximum audio-based distance.

A. General Remarks

The TSP is NP-hard, which implies that there is no known algorithm that calculates the exact result fast for large data sets. For example, the state of the art approach for the exact solution based on cutting planes and linear programming, would take more than 90 years of CPU time[3] for 24 978 cities.

Many heuristics have been proposed that approximate the correct result. We selected four of them for evaluation. We did not use an exact algorithm as the results would hardly be different but computation times would be much greater.

B. Domain-Specific Issues

A number of heuristic TSP algorithms require the distance measure d between the nodes to satisfy the triangle inequality $d(ac) \leq d(ab) + d(bc)$ for all triples a, b and c. In our scenario, the triangle inequality is not fulfilled by the similarity measure we used in our experiments.[4] On the audio similarity matrix constructed from our test data, it does not hold in about five percent of the cases when comparing randomly chosen direct and alternative edges.

According to [19], it can be proven that for nonmetric problems, it is impossible to construct an algorithm of polynomial complexity which finds tours whose length is bounded by a constant multiple of the optimal tour length (see also [20]).

C. Evaluated Algorithms

In the following sections, the algorithms we evaluated are described briefly. For a more detailed discussion, the interested reader is referred to the literature.

1) Greedy Algorithm: The first algorithm we evaluated is a popular simple greedy algorithm, e.g., [21]. The algorithm starts with no connected nodes. All edges are examined in order of increasing length. An edge is added to the initially empty set of edges if the resulting set of edges can still be combined to a valid tour. For m edges, this algorithm has a runtime of $O(m \log m)$, as the most expensive step is to sort the edges in ascending order.

2) Minimum Spanning Tree: This algorithm (e.g., [22]) was evaluated although it makes the assumption that the triangle inequality is fulfilled, which is not the case on the data we used. First, a minimum spanning tree is found with a standard algorithm (Kruskal algorithm) in $O(m \log m)$, with m being the number of edges. Afterwards, a depth-first search is performed on the minimum spanning tree, and a tour is constructed by connecting the nodes in the order they are first visited during the depth-first search. (For convenience, we call this whole algorithm *MinSpan* in this paper). On data that satisfies the triangle inequality *MinSpan* produces a tour that is guaranteed not to be longer than twice the optimum tour.

3) LKH: The LKH algorithm [20] is an optimized version of the Lin-Kernighan algorithm proposed by Lin and Kernighan in 1971. The LKH algorithm starts with a randomly generated tour and improves it incrementally by deleting λ edges from the route and recombining the remaining tour fragments in a more efficient way. In each step, sophisticated heuristics are used to choose λ and the edges to exchange.

The runtime of the LKH algorithm is approximately $O(n^{2.2})$, with n being the number of nodes.

4) One-Dimensional Self-Organizing Map (SOM): To assess if an algorithm based on clustering yields better results in terms of constancy of genre membership, also a SOM algorithm was evaluated.[5]

A SOM algorithm clusters the input data by assigning m input points to n points called *units*. During the (stochastic) clustering process, the position of each unit and the assignment of data points to units is refined iteratively. Usually, $n \ll m$, i.e., there are many more data points than units. SOMs have the property that they are able to project high-dimensional data into lower dimensional spaces while preserving distance relationships to a large extent. In our experiment, we train a one-dimensional (1-D) cyclic SOM, i.e., the units are arranged in a circular fashion. The one dimension corresponds to the position on the linear playlist.

Ideally, there would be as many units as tracks, so that after training, each unit would have exactly one assigned track. But since the runtime for training such a SOM is too long, we decided to use a recursive approach. After training the SOM, each track is assigned to the closest unit. If more than one track gets assigned to a specific unit, the algorithm is run recursively for the tracks that are assigned to this unit. The resulting smaller routes are combined in a greedy manner (cf. Algorithm 1). Before using the algorithm, the dimensionality is reduced by interpreting each column of the distance matrix as a vector and applying PCA on them. Only the first 30 components of the PCA-compressed data were used.

Algorithm 1: Recursive Algorithm based on SOM.

1: train a 1-D cyclic SOM with k units on the input points
2: for each input point, get the best matching unit of the trained SOM
{*calculate the k smaller tours ("subtours") recursively*:}
3: **for** each unit u_i $(i = 1 \ldots k)$ **do**
4: get the input points P_i belonging to the current unit u_i
5: if $|P_i > 1|$ build a tour through P_i recursively
6: store the point (or tour) in t_i
7: **end for**
{*combine the subtours, in a greedy manner}*:
8: break up each subtour at its longest edge
9: get all edges that could combine two *consecutive* subtours, store them in E
10: sort E in ascending order
11: **while** tour is not complete **do**
12: **if** next longer $e \in E$ can be part of a valid tour **then**
13: add it to the (still uncomplete) tour
14: **end if**
15: **end while**

[3]85 years to prove that the tour—found by a heuristic algorithm—is the shortest (http://www.tsp.gatech.edu/sweden/index.html).

[4]Even if it were, adding the penalty would break this condition.

[5]We used the Netlab toolbox, which is available at http://www.ncrg.aston.ac.uk/netlab/index.php.

V. Evaluation and Results

This section describes how the performance and differences between the audio-only approach and the combined approach were evaluated using the various TSP algorithms. First, we present the test collection on which the evaluation was performed. Then, several evaluation approaches are described. For each evaluation approach, its concept is outlined, and the results obtained are briefly discussed. Furthermore, we present setup and results of a user study which we conducted to assess the quality of the playlists generated by the combined approach.

For computing the similarity matrix, we used an in-house collection containing 2 545 tracks from 13 genres: A Cappella (4.4%), Acid Jazz (2.7%), Blues (2.5%), Bossa Nova (2.8%), Celtic (5.2%), Electronica (21.1%), Folk Rock (9.4%), Italian (5.6%), Jazz (5.3%), Metal (16.1%), Punk Rock (10.2%), Rap (12.9%), and Reggae (1.8%). Altogether, the collection consisted of tracks by 103 artists, whose names were used to query the web as described above. For each artist, at minimum eight tracks were in the collection. The maximum number of tracks by an individual artist was 61.

Regarding the runtimes of the algorithms, extracting the feature data and calculating the distance matrix are by far the most time consuming steps, taking several days. The runtimes of the TSP algorithms are much faster. On our evaluation collection, exemplary runtimes are 25 s for the MinSpan algorithm, 62 s for the simple greedy algorithm, 328 s for the SOM-based algorithm (taking additionally 245 s for the PCA step), and 132 s for the LKH algorithm (plus 36 s of data preprocessing).

A. Fluctuations Between Genres

To assess the genre transitions that occur most frequently along the path, Fig. 3 summarizes the genre memberships of pairs of tracks that follow immediately in the playlist for the LKH algorithm. In both cases, the most frequent genre transition is from Acid Jazz to Electronica. In our experiments, we use genre memberships as an indicator, as we assume that very similar pieces belong to the same genre.

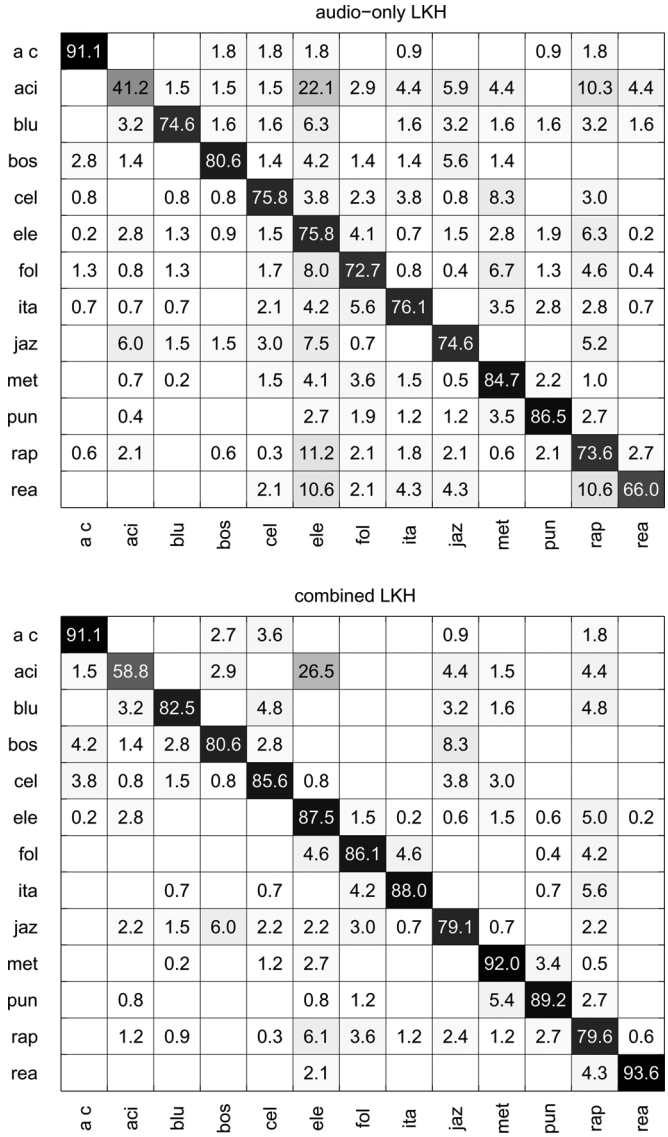


Fig. 3. Genre fluctuations for the best tour found by the LKH algorithm on the audio-only data (upper figure) and on the combined data (lower figure). Rows are the genres of the first track, and columns are the genres of the track following immediately in the playlist. Genres are: A Cappella, Acid Jazz, Blues, Bossa Nova, Celtic, Electronica, Folk Rock, Italian, Jazz, Metal, Punk Rock, Rap, and Reggae, denoted by the first three characters.

B. Shannon Entropy

To estimate how "locally coherent" a playlist is, the entropy of the genre distribution was calculated on short sequences of the playlists: It was counted how many of n consecutive tracks belonged to each genre. The result was normalized and interpreted as a probability distribution, on which the Shannon entropy was calculated. The Shannon entropy is defined as

$$H(x) = -\sum_{x} p(x) \log_2 p(x) \tag{3}$$

with $\log_2 p(x) = 0$ if $p(x) = 0$.

In Fig. 4, the entropy values for $n = 2 \ldots 12$ are given, averaged over the whole playlist (i.e., each track of the playlist was chosen once as the starting track for a sequence of length n). 12 was chosen as the maximum length because a typical album contains about 12 tracks.

The most important finding is that even the worst-performing algorithm (SOM) yields better results on web-enhanced data than the best (LKH) on audio-only data.

C. Long-Term Consistency

Complementary to the short-term development measured by the entropy, it is also interesting to assess the long-term development of the generated playlists. To this end, Fig. 5 shows the distribution of each genre over the whole playlist. We decided to include the result with the worst overall impression with respect to this aspect (top), the best result that has been achieved on audio data (middle), and the best overall result (bottom).

Obviously, the playlist generated by the simple greedy algorithm on audio data is highly fragmented. In the diagram for the MinSpan algorithm, a diagonal-like arrangement becomes apparent. The genres tend to concentrate at certain regions. This demonstrates that our approach is able to produce reasonable playlists based purely on audio content. These results can be

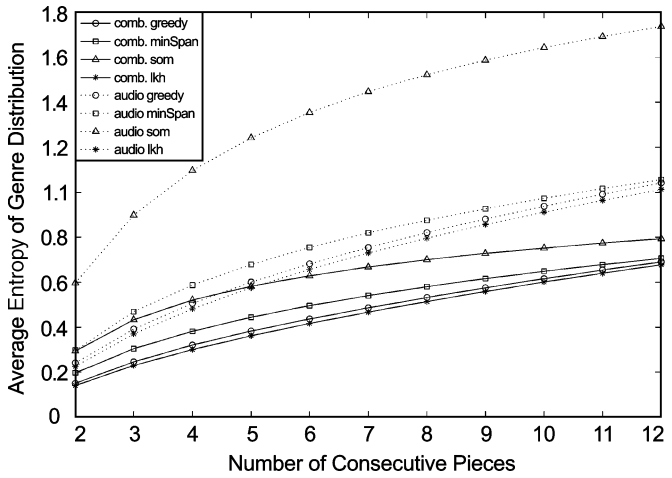


Fig. 4. Average entropy values for short sequences (length 2..12) of the playlists. The experiments based on audio only are shown as dotted lines, the combined approaches as solid lines. Same TSP algorithms have same markers.

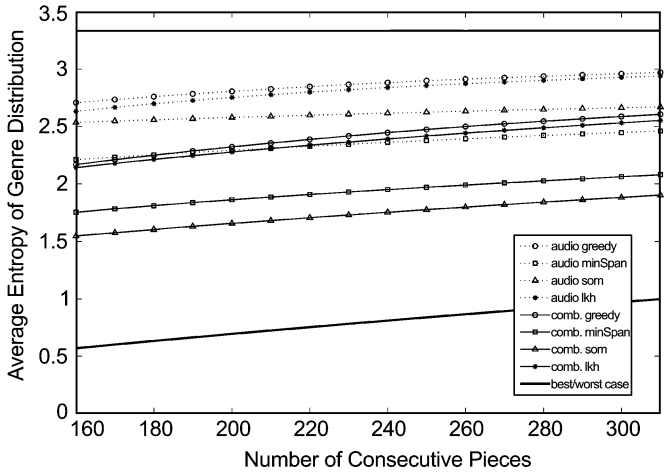


Fig. 6. Long-term genre entropy values for the range from 160 to 318 tracks, which corresponds to angles between 22.5 and 45 degrees on the wheel. Same markers and line styles as in Fig. 4.

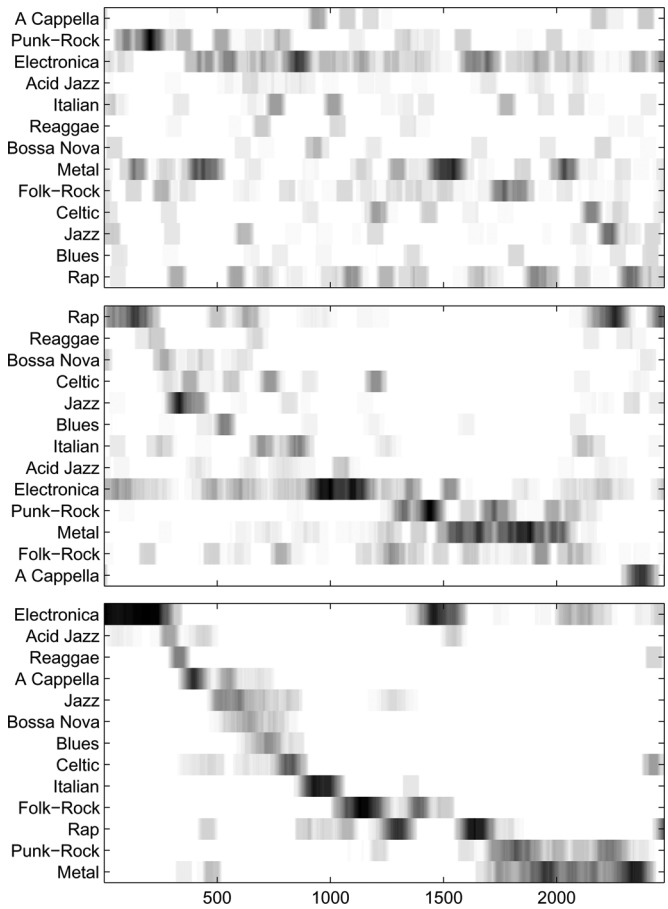


Fig. 5. Long-term distribution of the genres for the playlists generated by the simple greedy algorithm on audio similarity data (top), the MinSpan algorithm on audio similarity data (middle), and the SOM algorithm on combined data (bottom). For better visibility, the values are blurred using a low-pass filter: for each track, it is counted how many of the following 75 tracks in the playlist have a particular genre. Full white represents 0, full black 75. In each diagram, the genres are ordered by the index where most pieces of that genre are accumulated. This point was determined by applying a window of length n, where n is the number of pieces in that genre. Thus, in the optimum case a playlist would tend to result in a diagonally descending sequence of black bars. Note that there is no canonical sequence of genres.

further improved by using our audio-and-web-combining similarity measure, as can be seen by the outcome of the SOM algorithm. This is at least circumstantial evidence that we are able to arrange music in a manner that is understandable to humans.

To quantify the visual impressions of the diagrams in Fig. 5, we use the same measurement as in Section V-B at a longer timescale. This time, we do not want to assess how well tracks match that are played in a sequence, but we want to know how clearly defined the type of music is in a certain *region* of the wheel. For this purpose, we assume that a user would expect the music to be consistent in an angle of about 22.5 to 45 degrees. Fig. 6 gives the values obtained on our test collection for this range, together with the possible minimum and maximum value for each region size.[6] We assume that lower entropy values indicate a more intuitive arrangement of the music since they represent fewer genre transitions. This assumption is confirmed when comparing the entropy values to the diagrams in Fig. 5 (and those figures not depicted here). Furthermore, it is interesting to note that in the long term genre distribution, the best audio-only based approach (i.e., audio minSpan) yields slightly better results than two of the algorithms that ran on the combined data. This gives an indication that the choice of the TSP algorithm is as important as improving the data it runs on.

D. Alternative Music Collection

To study the robustness of our approach with respect to different music collections, we compiled a second, larger collection which was also used for the user study described in the next section. It contains 3456 tracks (by 339 artists) assigned to 7 general, quite evenly distributed genres: Classical (14.7%), Dance (15.0%), Hip-Hop (14.5%), Jazz (13.6%), Metal (14. 9%), Pop (11.6%), and Punk (15.6%). The minimum number of tracks per artist is 1, the maximum 317.

The evaluation results of the second collection are largely comparable to those obtained with the first one. The number of genre transitions present in playlists generated from the second

[6]Note that the minimum depends on the number of pieces in each genre and on each genre appearing as a closed block in the playlist, but not on the order in which the genres are arranged.

TABLE I
RESULTS OF BOTH EVALUATION MEASURES OF THE USER STUDY. BOLD FACED ENTRIES INDICATE SUPERIOR RATING FOR OUR TSP-APPROACH

playlist no.	1	2	3	4	5	6	7	8	9	10	$\sum$
$\sum tsp$	48	36	26	27	38	41	36	50	50	23	375
$\sum gen$	34	16	42	36	33	36	28	37	39	40	341
ratio	**1.4**	**2.3**	0.6	0.8	**1.2**	**1.1**	**1.3**	**1.4**	**1.3**	0.6	1.1
$\#_{pos} : \#_{neg}$	**10:0**	**10:0**	0:9	2:5	**5:1**	**4:0**	**6:2**	**8:0**	**8:0**	0:9	53:26

collection is consistently lower than that of the first one. Also, the values for the long-term entropy are always lower for the second collection. This is no surprise, since the second collection contains three times as many artists as the first but only half as many genres.

E. User Study

To get an impression of the "usefulness" of the resulting playlists, we carried out a small user study on 10 test persons $j \in [1, 10]$, using the second test collection. We created a large playlist with the MinSpan algorithm and the combined similarity measure. From this, we extracted ten seed tracks by randomly choosing a starting point on the "wheel" and consecutively selecting tracks at intervals of 36 degrees. From each seed track we created two playlists—one by sequentially adding the next nine tracks from the "wheel" (denoted by tsp_i in the following), the other by randomly adding nine other tracks *from the same genre* as the seed track (gen_i). Each test person had to rate each playlist with respect to overall musical consistency on a scale ranging from 1 ("totally inconsistent") to 5 ("completely smooth transitions"). Furthermore, is was possible to add remarks to every playlist, from which we gained deeper insights into the ratings.

To evaluate the resulting ratings, we compute two measures for each pair of playlists tsp_i and gen_i. We obtain the first measure by simply summing up the rating scores for both types of playlists over all users and calculate the ratio of the tsp-based score and the gen-based score, i.e., $\sum_j \mathrm{tsp}_{i,j} / \sum_j \mathrm{gen}_{i,j}$. For the second measure, we calculate the difference $\mathrm{tsp}_{i,j} - \mathrm{gen}_{i,j}$, for each pair and test person, and count the number of positive differences $\#_{\mathrm{pos},i}$ and negative differences $\#_{\mathrm{neg},i}$ over all test persons. The second measure is then the ratio $\#_{\mathrm{pos},i} : \#_{\mathrm{neg},i}$. Results from our evaluation can be found in Table I.

It can be seen that in seven out of ten cases, our TSP-based playlist approach yields better user ratings than a straight-forward genre-based shuffling approach. Noticeably, the difference is less distinct for the first measure (375:341) than for the second (53:26). This can be explained by the fact that a few unexpected tracks in some playlists led to very low ratings, which is also evidenced by the additional user comments. On the second measure, this effect has no impact.

F. Qualitative Evaluation

For subjective evaluation, a Java applet was programmed that enables the user to quickly browse through the circular playlist (cf. Fig. 1). As input device, the Griffin PowerMate[7] was used. Non-representative small-scale usability and listening tests revealed that subjects generally liked the idea of such a sound player.

As for the quality of the playlists, the findings of the conducted user study (cf. Section V-E) were largely confirmed. In general, consecutive tracks were perceived as sounding similar. However, users got the impression that the path is not always fully straightened, i.e., musical regions are revisited after leaving them.

As for the user interface, first impressions were quite positive. The subjects remarked that the user interface is very intuitive and its handling extremely easy. Users liked the feel of spinning the PowerMate, but could also imagine to use the user interface on Apple's "iPod". Since the PowerMate can be pressed, one user suggested a "scanning" function to skip 10 s of the current track when pressing it. A negative remark was that genres containing only a few tracks are quite difficult to locate, another one that the device is not usable when the aim is to find a specific track in a collection. On the whole, users found the sound player especially useful for listening to music in situations when they cannot pay much attention to the handling of the device (e.g., while doing sports).

G. Summary of the Evaluation Results

We used several different measures to evaluate the TSP algorithms and our two similarity measures. The results of these evaluations are summarized in the following.

- Most importantly, all TSP algorithms provided better results with respect to our playlist evaluation criteria when using the web based extension. However, the time consuming step of the web retrieval must be taken into account.
- We observed that the combined similarity measure reduces the number of unexpected placements of tracks in the playlist, i.e., spontaneous transitions between tracks from different genres.
- The LKH algorithm and the simple greedy algorithm have the best small-scale genre entropy values. On the other hand their large-scale genre distributions are quite fragmented. The SOM-based algorithm yields the highest entropy values, but the least fragmented long-term genre distributions. The MinSpan algorithm was in the middle field regarding the entropy values. The overall genre distribution was inferior to the one of the SOM-based approach.

The MinSpan algorithm and the SOM-based approach produce better overall genre distributions than the LKH and the simple greedy algorithm. The most likely reason for that is that locality constraints are regarded: in the SOM algorithm the overall route is built by concatenating several local routes, and in the MinSpan algorithm locality constraints are regarded as the algorithm constructs the route by a depth-first search on a minimum spanning tree.

As a final recommendation, it follows that either the SOM-based algorithm or the MinSpan algorithm seem to be favorable. Furthermore, we can state that incorporating web-based metadata is beneficial.

VI. CONCLUSION AND FUTURE WORK

We presented a new approach to conveniently access the music stored in mobile sound players. The whole collection

[7]http://www.griffintechnology.com/products/powermate.

is ordered in a circular playlist and thus accessible with only one input wheel. Consecutive tracks are aimed to be consistent and maximally similar on average, thus ideally each track can be chosen as the starting point of a locally consistent playlist. Several algorithms were assessed with respect to their ability to produce such playlists.

We implemented a demonstration application for a single-dial browsing device. Such a device offers a new way to access music collections, as tracks are arranged according to the underlying similarity measure. We suggest two different similarity measures—one relying on timbre information, the other on a combination of timbre and community metadata gathered from artist related web pages. It is worth mentioning that we present one of the first works on combining the two research areas of audio-based track similarity and community-based artist similarity in a useful way.

The example application triggers positive reactions and provokes the user to play around with it. We used two music collections for evaluation, one containing about 3 500 tracks from very general genres, the other one containing more than 2 500 pieces from a mixture of quite specific and general genres. Due to this large number of tracks in the test collections, with the current device it is not possible to precisely select a desired piece. It is still an open question how to deal with this situation. One possibility would be to make only tracks selectable that are representative for a region. Alternatively, zooming or hierarchical structuring techniques could be applied.

Another drawback of the current version of the music player is that the user does not know in advance which region on the wheel contains which style of music. First steps towards a solution can be found in [23], where we extended the player by visualizing distributions of meta-data (e.g., genre) around the wheel to facilitate browsing the collection.

On the whole, the results of the conducted user study showed that people perceive the playlists generated with our audio-and-web-combining similarity measure more consistent than playlists generated by randomly selecting pieces within a given genre.

REFERENCES

[1] T. Pohle, E. Pampalk, and G. Widmer, "Generating similarity-based playlists using traveling salesman algorithms," in *Proc. 8th Int. Conf. Digital Audio Effects (DAFx-05)*, Madrid, Spain, Sep. 20–22, 2005, pp. 220–225.

[2] M. Alghoniemy and A. Tewfik, "A network flow model for playlist generation," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME'01)*, Tokyo, Japan, Aug. 22–25, 2001 [Online]. Available: http://citeseer.ist.psu.edu/alghoniemy01network.html

[3] J.-J. Aucouturier and F. Pachet, "Scaling up music playlist generation," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME'02)*, Lausanne, Switzerland, Aug. 26–29, 2002.

[4] B. Logan, "Content-Based playlist generation: exploratory experiments," in *Proc. Int. Conf. Music Information Retrieval (ISMIR 2002)*, Paris, France, Oct. 13–17, 2002.

[5] B. Logan and A. Salomon, "A music similarity function based on signal analysis," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME'01)*, Tokyo, Japan, Aug. 22–25, 2001.

[6] G. Tzanetakis, "Musescape: an interactive content-aware music browser," in *Proc. 6th Int. Conf. Digital Audio Effects (DAFx-03)*, London, U.K., Sep. 8–11, 2003.

[7] M. Schedl, E. Pampalk, and G. Widmer, "Intelligent structuring and exploration of digital music collections," *e&i—Elektrotechnik und Informationstechnik*, vol. 122, no. 7/8, Jul./Aug. 232–237, 2005.

[8] T. Kohonen, "Self-organizing formation of topologically correct feature maps," *Biolog. Cybern.*, vol. 43, pp. 59–69, 1982.

[9] E. Pampalk, A. Rauber, and D. Merkl, "Using smoothed data histograms for cluster visualization in self-organizing maps," Proc. Int. Conf. Artifical Neural Networks (ICANN'02). Madrid, Spain, Springer, Aug. 2002, pp. 871–876.

[10] M. Goto and T. Goto, "Musicream: new music playback interface for streaming, sticking, sorting, and recalling musical pieces," in *Proc. Sixth Int. Conf. Music Information Retrieval (ISMIR'05)*, London, U.K., Sep. 2005.

[11] E. Pampalk, A. Rauber, and D. Merkl, "Content-based organization and visualization of music archives," Proc. ACM Multimedia. Juan les Pins, France, ACM, Dec. 1–6, 2002, pp. 570–579.

[12] T. Pohle, "Extraction of Audio Descriptors and Their Evaluation in Music Classification Tasks" Master's thesis, Technische Univ. Kaiserslautern, Austrian Res. Inst. Artificial Intelligence (ÖFAI), Deutsches Forschungszentrum für Künsieetliche Intelligenz (DFKI), , 2005 [Online]. Available: http://kluedo.ub.uni-kl.de/volltexte/2005/1881/

[13] J.-J. Aucouturier, F. Pachet, and M. Sandler, ""The way it sounds": timbre models for analysis and retrieval of music signals," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1028–1035, Dec. 2005.

[14] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *Proc. 1st Int. Symp. Music Information Retrieval*, 2000 [Online]. Available: http://citeseer.ist.psu.edu/logan00mel.html

[15] J.-J. Aucouturier and F. Pachet, "Music similarity measures: what's the use?," Proc. 3rd Int. Symp. Music Information Retrieval IRCAM. Paris, France, Oct. 13–17, 2002, pp. 157–163.

[16] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[17] J.-J. Aucouturier and F. Pachet, "Improving timbre similarity: how high is the sky?," *J. Negative Results in Speech and Audio Sci.*, vol. 1, no. 1, 2004.

[18] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inform. Process. and Manag.*, vol. 24, no. 5, pp. 513–523, 1988.

[19] S. Sahni and T. Gonzales, "P-complete approximation algorithms," *J. Assoc. Comput. Mach.*, vol. 23, pp. 555–565, Jul. 1976.

[20] K. Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. Roskilde, Denmark, Roskilde Univ., DATALOGISKE SKRIFTER (Writings on Computer Science), vol. 81, 1998.

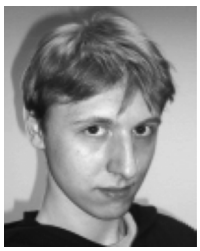
[21] D. Johnson and L. McGeoch, "The traveling salesman problem: a case study in local optimization," in *Local Search in Combinatorial Optimization*. New York: Wiley, 1997, pp. 215–310.

[22] S. Skiena, The Algorithm Design Manual Dept. Comput. Sci., State Univ. New York, Stony Brook, Springer-Verlag. New York, 1997.

[23] M. Schedl, T. Pohle, P. Knees, and G. Widmer, "Assigning and visualizing music genres by web-based co-occurrence analysis," in *Proc. 7th Int. Conf. Music Information Retrieval (ISMIR'06)*, Victoria, Canada, Oct. 2006.

Tim Pohle has studied musicology and computer science. He received the Dipl.-Inf. degree (equivalent to an M.Sc. degree in computer science) from the Technical University of Kaiserslautern, Kaiserslautern, Germany. He is currently pursuing the Ph.D. degree at Johannes Kepler University Linz, Austria.

He is a Research Assistant in the field of music information retrieval, with a special emphasis on audio-based techniques, at Johannes Kepler University.

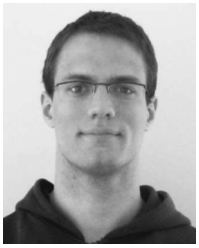


Peter Knees received the diploma in computer science, with a thesis on "Automatic Classification of Musical Artists based on Web-Data". He is currently pursuing the Ph.D. degree at Johannes Kepler University Linz, Austria, with a focus on music information retrieval. He is also studying psychology at the University of Vienna, Vienna, Austria.

Since February 2005, he has been a Project Assistant at the Department of Computational Perception, Johannes Kepler University.

Markus Schedl graduated in computer science from Vienna University of Technology, Vienna, Austria.

Since 2003, he has been studying International Business Administration at the Vienna University of Economics and Business Administration. Since 2004, he has been working on his doctoral thesis in Computer Science at the Department of Computational Perception. His main research interests include Web Mining, (Music) Information Retrieval, Information Visualization, and Intelligent User Interfaces.



Elias Pampalk received the Ph.D. degree, with a thesis on computational models of music similarity and their application in music information retrieval, from the Vienna University of Technology, Vienna, Austria, in March 2006.

During his doctoral studies, he worked at the Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, supervised by G. Widmer. He is currently with the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan.

Gerhard Widmer received the M.Sc. degrees from the University of Technology, Vienna, Austria, and the University of Wisconsin, Madison, and the Ph.D. degree in computer science from the University of Technology, Vienna.

He is a Professor and Head of the Department of Computational Perception at Johannes Kepler University, Linz, Austria, and Head of the Intelligent Music Processing and Machine Learning Group at the Austrian Research Institute for Artificial Intelligence, Vienna, Austria. His research interests include machine learning, pattern recognition, and intelligent music processing.

Dr. Widmer was awarded one of Austria's highest research prizes, the START Prize, in 1998 for his work on AI and music.