A Graph-Based Method for Playlist Generation

Debora C. Correa[1], Alexandre L. M. Levada[2] and Luciano da F. Costa[1]

[1] Instituto de Fisica de Sao Carlos, Universidade de Sao Paulo, Sao Carlos, SP, Brazil
[2] Departamento de Computacao, Universidade Federal de Sao Carlos, SP, Brazil
deboracorrea@ursa.ifsc.usp.br, alexandre@dc.ufscar.br,
luciano@ifsc.usp.br

Abstract. The advance of online music libraries has increased the importance of recommendation systems. The task of automatic playlist generation naturally arises as an interesting approach to this problem. Most of existing applications use some similarity criterion between the songs or are based on manual user interaction. In this work, we propose a novel algorithm for automatic playlist generation based on paths in Minimum Spanning Trees (MST's) of music networks. A motivation is to incorporate the relationship between music genres and expression of emotions by capturing the presence of temporal rhythmic patterns. One of the major advantages of the proposed method is the use of edge weights in the searching process (maximizing the similarity between subsequent songs), while Breadth-First (BF) and Depth-First (DF) search algorithms assume the hypothesis that all the songs are equidistant.

Keywords: playlist, rhythm, graphs, search algorithms.

1 Introduction

With the dissemination of online resources with music content, music recommendation systems have received much attention. Indeed, the sometimes manual and time-consuming selection task of music playlists can be replaced by automatic algorithms. Such algorithms can generate the playlist according to the user's music preferences or through some defined similarity criterion between the songs.

We can relate three main important aspects of a playlist: the individual songs themselves, the order in which they are played, and the size of the playlist. In the literature, we can find earlier efforts concerning the automatic generation of musical playlists [1, 10, 7, 9]. Most of these approaches are based on collaborative filtering techniques, audio content analysis, and require a manually labeled database or the analysis of metadata.

In this scenario, our contribution is to propose a novel and unsupervised playlist algorithm, avoiding possible noise due to the user's subjectiveness. Besides, a motivation is the possibility to associate music genres to the presence of temporal patterns in the rhythm as a way to express notions of emotion. According to [12], regular and smooth rhythmic patterns indicate expressions like happiness, joyness or peacefulness. Irregular and complex rhythms indicate

9th International Symposium on Computer Music Modelling and Retrieval (CMMR 2012)
19-22 June 2012, Queen Mary University of London

expressions like amusement, tension or uneasiness, while fluent rhythms indicate feelings like happiness, gracefulness or dreamy. Thus, in our playlists the songs are rhythmically related and, therefore, emotionally linked.

The algorithm is performed on a Minimum Spanning Tree (MST) extracted from music networks. Although we have not performed a conclusive user evaluation of the subjective quality of the the playlists, this investigative work relies on the properties of the MST, reflecting the overall characteristics of the playlists. The similarity criterion used to build the music networks is based on the cosine distance between feature vectors of the songs. Thus, the playlist can be easily adapted to other types of musical feature and similarity metrics [13].

The remaining parts of the paper are organized as follows: section 2 describes the database, the methodology to construct the music networks, and the extraction process of the note duration dynamics; section 2.4 presents the proposed algorithm for automatic generation of the music playlists and a discussion of its characteristics. Finally, section 3 contains the main conclusions.

2 Constructing music graphs

2.1 The database

Our database consists of four musical genres with seventy samples each: blues, *mpb* (*Brazilian popular music*), reggae and rock. Our motivation for choosing these four genres is the availability of MIDI samples in the Internet with considerable quality and the different tendencies they represent.

MIDI format is simpler to analyse than audio files, since all voices are separated in tracks. However, as it is a symbolic representation, MIDI allows a clear analysis of the involving music elements, in opposite to audio files in which all information is mixed together. To read a MIDI file, we used the Sibelius software and the free Midi Toolbox for Matlab computing environment [11]. This toolbox provides a note matrix representation, with information like relative duration (in beats), MIDI pitch, and others. The note value is represented in this matrix through relative numbers. To deal with possible fluctuations in tempo, we unset the "Live Playback" option in Sibelius. In this way, the note values in the MIDI file respect their relative proportion (e.g, the eighth note is always 0.5).

As we want to generate rhythm-based similarity playlists, we propose a similarity criterion based on the temporal sequence of the note values present in the percussion track. In this work, the instrumentation is not considered. If two or more note events occur in the same beat, the median duration of them is taken. To clarify this concept, Figure 1 shows the first measures of the percussion track of the music *Who can it be now?* (Men at Work).

Part of the matrix representation of the notes values of the third measure is presented in Table 1. Taking the median value for events occurring at the same beat, the final vector with note values is: [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]. The note vector of the whole percussion is computed for each song in the database. All this process can be performed automatically.

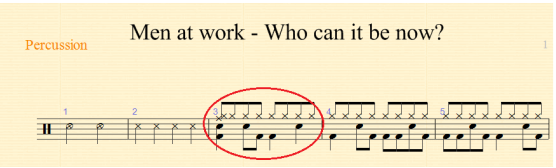

Fig. 1. Example of a percussion track.

Beat	8	8	8	8	8.5	9	9	9.5	9.5	10	10	10.5	11	11	11.5
Relative Duration	0.5	0.5	1	1	0.5	0.5	0.5	0.5	0.5	0.5	1	0.5	0.5	1	0.5

Table 1. Matrix representation of first measure of the percusion in Figure 1.

2.2 Markov modeling for note value dynamics

Markov chains establish a conditional probability structure in which the probability of an event n depends on one or more past events $(n-1, n-2...)$[5]. The order of the chain is determined by the number of past events taken into consideration. A first order Markov chain models the dependency between the current event and its predecessor. Similarly, a second order Markov chain takes into account the two subsequent past events. Therefore, a nth-order Markov chain will set up a transition matrix of $n+1$ dimensions, in which each entry gives the conditional probability of an event, based on its previous n states. We propose that the events to be modeled in the Markov chains be the note values extracted from the percussion track of the songs. For each note vector, we compute the first and second order probability transition matrices.

In order to reduce data dimensionality, we performed a preliminary analysis of the relative frequency of note values and pairs of note values concerning all the songs, in a way that extremely rare transitions were discarded[1]. We considered 18 different note values in the first order Markov chain (isolated note values) and 167 different pairs of note values for the second order Markov chain. Therefore, the first order probability transition matrix is 18 (rows) x 18 (columns), indicating the conditional probability that a note value i is followed by a note value j. The second order probability transition matrix is 167 (rows) x 18 (columns), indicating the conditional probability that a pair of note values represented in row i is followed by a note value in column j. Both matrices are treated as a 1 x 324 and 1 x 3006 (167 * 18) feature vector, respectively. To form the final feature vector of each song, we concatenate both feature vectors (3330 dimensions).

2.3 Music Networks and Playlist Generation

A complex network is a graph that exhibits non-trivial topological structure between its elements [6]. A graph can be represented by vertices, edges (links),

[1] We count how many times each note duration appears considering all songs and discard the ones that occurred less than 0.1% of times

and weights associated to the links. In this case, each edge has the form $w(i, j)$, expressing the weight $w(i, j)$ in the connection from vertex i to vertex j.

Each song is represented by a vertex. The weight of the link between two songs is expressed by the distance between their feature vectors. We considered the cosine distance, although many alternative distance metrics can be used [13]. However, it may be difficult to analyse intricate structures if the network is full-connected. Therefore, the motivation is to construct the playlist under the properties of a MST. Different distance metrics leads to different MSTs and, consequently, to different playlists. From the point of view of playlist generation, these variations are profit, allowing many possibilities of sequences of music.

The proposed music network allows a straightfoward automated recommendation scheme through a simple playlist generation algorithm, based on paths on a MST. Since the networks are built in a completely unsupervised manner, the user labeling is not required. The idea is to generate a playlist according to a similarity criterion, assured both by the network construction process, and by the MST properties (connectivity among all vertices and minimum dispersion).

Suppose one wants to perform a search for similar vertices in a music network. A possible solution is to apply a simple weighted random walk, a BF search algorithm (broadcast) or even a degree-based searching on the original graph [3]. However, in these situations, one may reach undesirable vertices once there is no way to assure that transitions between vertices are smooth in the sense that each visited vertex is somehow similar to the previous ones.

We propose that a MST can be used in the definition of a new search strategy, named the jumping walk algorithm. In few words, a MST is a minimum-weight acyclic connected subgraph in which there is only one single path from any vertex A to any other vertex B. Moreover, due to its properties, if vertex A is linked to vertices B and C by using a distance metric weighted edge, it means that both B and C are the two most similar ones regarding vertex A (considering the tree and not the original graph). Thus, visiting nearest neighbors in a MST will produce a sequence of vertices in which the next element is the most similar to the previous ones in the tree. We used the Prim's algorithm to generate a MST from the full-connected music network [8, 6].

2.4 The Jumping Walk Algorithm

The searching strategy is based on walks on MST's without repetition of vertices. A simple approach for playlist generation is to perform a breadth-first search in the resulting MST. The basic procedure is to form a list of songs by first visiting all vertices that are a distance one from the starting node, then visiting all vertices that are a distance two and so on until we reach a maximum depth. Although simple and functional, this approach has some drawbacks. First, it completely ignores the edge weights, assuming the hypothesis that all vertices are equidistant in the feature space, which is not reasonable. And second, the set of vertices that are located a distance d from a starting node are not linked in the MST, which means that these samples may not be close enough in the feature space. This effect is even worse for large values of d, where the equidistant

samples can be from totally different clusters. However, when walking through a MST, an unexpected situation may occur if we hit a leaf of the tree (dead-end path). The proposed strategy deals with such situation by jumping to the nearest unvisited vertex and restarting a new walk, as described in the following.

The jumping walk algorithm consists of a sequence of walks in the MST of a music network. The algorithm starts by choosing a random initial vertex representing a specific music genre, given as input by the user. After that, a sequence of vertices is defined by the continuous choice of the nearest unvisited neighbor (minimum weight edge) to be the next song. If the selected song is a leaf (that is, there is no unvisited neighbor), then we check all the vertices that are a distance $d = 2, 3, 4...$ until we find an unvisited vertex. In case of more than one option, we perform a jump to the nearest one according to the edge weights (this is equivalent to finding the shortest-path between the leaf and an unvisited vertex using the Djikstra algorithm). Even though in this case there is a jump, that is, a discontinuity in the sequence, the MST properties guarantees that the next song is the most similar song in the tree that was not played yet. Overall, after each jump (from a leaf of the tree), a new walk is started.

A potential problem with this method is the generation of playlists with M songs, where $M \approx N$ (N is the number of songs). With the rule of walking to the nearest neighbor (in terms of minimum weight), some branches of the MST may be neglected, which may cause large jumps as the number of unvisited vertices aproximates zero. Two ways to avoid this problem are: 1) having a sufficient large N (e.g., 100000); 2) stopping the walks whenever the number of visited vertices reaches an upper bound (e.g., $0.75 * N$).

Figure 2 illustrates the jump process. Suppose that we start in A, the sequence of visited vertices before we reach a leaf will be A, B and C. After that, a jump to the nearest vertex is performed. Note that the edges AB and BC are in the minimum path from C to D and also in the minimum path from C to E. Therefore, the next selected song (D in this case) will be the nearest to both C (position i in the list) and A (position $i - 2$ in the list).

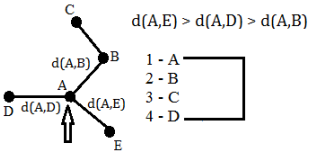

Fig. 2. An illustration of the jumping walk process.

Figure 3 shows the MST for the music network using the cosine distance between the vertices attributes. Many songs of the same genre in branches of the a same subtree. Applying the jumping walk algorithm and using the song number 39 as initial vertex, the top twenty five songs of the playlist are:

39 - *(blues)* Johnny Winter - Good morning little schoolgirl 50 - *(blues)* Stevie Ray Vaughan - Cold shot 62 - *(blues)* Stevie Ray Vaughan - Tell me 23 - *(blues)* Delmore Brothers - Blues stay away from me 203 - *(reggae)* Third World - Now that we've found love 25 - *(blues)* ElvesPresley - A mess of blues 44 - *(blues)* Ray Charles - Born to the Blue 12 - *(blues)* Barbra Streisand - Am I Blue? 38 - *(blues)* John Lee Hooker - One bourbon one scotch one beer 16 - *(blues)* Boy Williamson - Dont start me talking 18 - *(blues)* Boy Williamson - Keep it to yourself 152 - *(reggae)* Bob Marley - I shot the sheriff 36 - *(blues)* John Lee Hooker - Boom boom boom 3 - *(blues)* AlbertKing - Stormy monday 34 - *(blues)* Jimmie Cox - Before you accuse me 190 - *(reggae)* Natiruts - Liberdade pra dentro da cabeca 5 - *(blues)* BB King - Dont answer the door 193 - *(reggae)* Nazarite Skank 263 - *(rock)* Rolling Stones - Angie 239 - *(rock)* Metallica - Fuel 245 - *(rock)* Metallica - Sad but true 29 - *(blues)* Freddie King - Hide away 170 - *(reggae)* Cidade Negra - Eu fui eu fui 40 - *(blues)* Koko Taylor - Hey bartender 59 - *(blues)* Stevie Ray Vaughan - Manic depression

The resulting playlist is based on the temporal patterns of the note values in the percussion. Hence, the algorithm will not necessarily generate a sequence of songs belonging to the same genre, but, instead, a sequence in which the songs are similar according to the note value patterns, which can indicate a relationship between subject aspects such as mood and emotion.

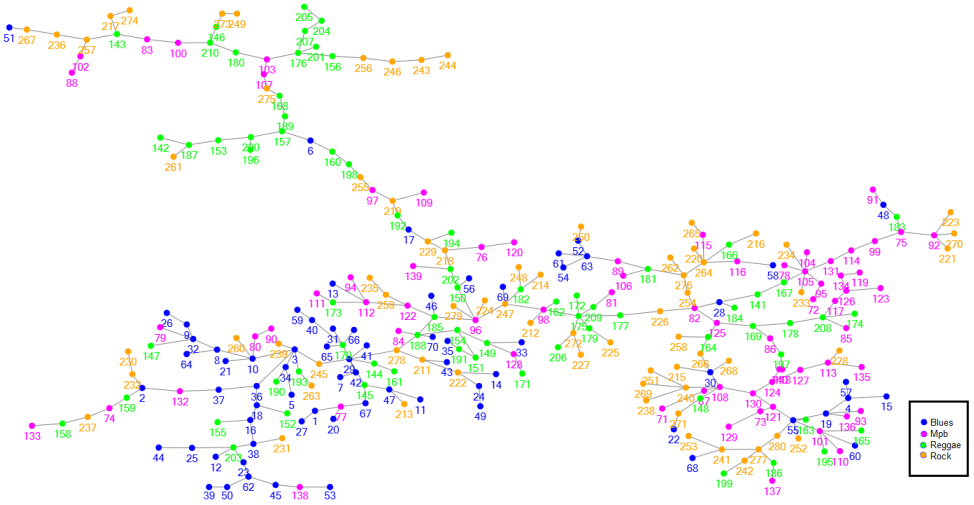

Fig. 3. MST for the network built with the cosine distance between vertices.

Different MSTs (derived from distincted music networks) lead to different playlists. In fact, multiple executions of PRIM's algorithm on the same network do not guarantee that the obtained MST's will be the same. For the purpose of playlists generation, this is a positive characteristic (controlled variability).

In a full-connected network, each song is connected to all other songs. If instead we link a song to its k nearest songs, we will have a k-regular network (digraph), since only the k nearest songs are linked together, avoiding long distance inter-genre connections. With the purpose of comparing the proposed approach for playlist generation with the BF search algorithm, we empirically

chose $k = 10$. In all experiments, when we refer to the BF search in the digraph, we mean that the search was performed in the 10-regular digraph.

For the first 100 songs in different playlists (using song 39 as the initial vertex), we analyzed the temporal aspects concerning the distance of subsequent songs as well as the inter-genre dynamics (Figure 4). We compared the JW algorithm with two different variations of the BF search algorithm: BF on the MST; BF on the 10-regular digraph. We can observe different behaviors depending on the adopting strategy. The mean distance is also presented for each situation. It shoud be noted that the JW algorithm produces the minimum mean inter songs distance. This means that, in average, the selected songs are more closely related than in the other methods. We analyzed the pairs of subsequent music genres for all the obtained playlists. The number of transitions between genres for each method were: JW = 57, BF on the MST = 60, BF on the 10-regular digraph = 73, indicating that the playlist produced by the JW algorithm was able to minimize the transitions between genres. Eventually, the distance between a pair of songs on a digraph may be undefined (as occured in Figure 4.)

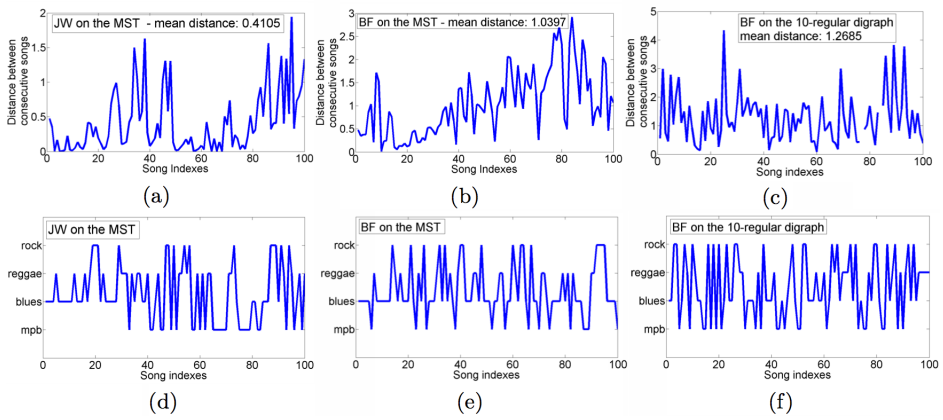

Fig. 4. Distances between subsequent songs and inter-genre interaction for the first 100 songs in the playlist obtained for the JW algorithm and BF searches.

3 Final remarks and ongoing work

We proposed a novel automatic algorithm for rhythm-based playlist generation based on the minimum spanning trees of music networks, It does not require user labeling, and can be easily implemented and adapted to other musical features.

In summary, the main contributions of the proposed method are: 1) the distance between consecutive songs in the playlist is minimized compared to BF search; 2) it reduces the number of abrupt transitions between genres; 3) songs from different genres are placed together if they have common rhythmic

patterns, which allows a controlled variability; 4) songs placed closely together also tend to have similar subjective aspects of mood.

Besides, BF searches on digraphs provide a much higher degree of freedom in the sense that we would face more unreliable inter-genre connections. However, we cannot guarantee the preference for the playlist obtained by MST paths or by BF searches on digraphs. It will mainly depend on the user's evaluation. What we can say is that, according to the temporal aspects presented in Figure 4, the MST-based playlists, in average, maximize the similarity between subsequent songs, given a specific similarity metric and a musical feature.

Future works include the evaluation of the playlists by systematic tests in the audience. With this initial investigation we hope to define a MIR application that performs queries in a database using as input a song that is not necessary stored in it. Furthermore, the use of shortest-path trees (Dijkstra algorithm) may bring additional insights to the relationship between genres and emotions.

Acknowledgments. Debora C Correa is grateful to FAPESP (2009/50142-0) for financial support, Alexandre L. M. Levada is grateful to CNPq (475054/2011-3) financial support, and Luciano da F. Costa is grateful to CNPq (301303/06-1 and 573583/2008-0) and FAPESP (05/00587-5) for financial support.

References

1. Andric, A., Haus, G.: Automatic playlist generation based on tracking user's listening habits. Multimedia Tools and Applications Journal. 29, Issue 2 (2006)
2. Roads, C. : The Computer Music Tutorial. MIT Press, Massachusetts (1996)
3. Easley. D, Kleinberg, J.: Networks, Crowds and Markets: Reasoning About a Highly Connected World. Cambridge University Press, Cambridge (2010)
4. Miranda, E: Composing with computers. Focal Press, Oxford (2001)
5. Isaacson, D. L., Madsen, R. W.: Markov chains, theory and applications. Krieger Pub Co, Malabar (1976)
6. Clarck, J., Holton, D.A: A First Look at Graph Theory. Word Scientific, Singapore (1991)
7. Alghoniemy, M., Tewfik, A.H.: A network flow model for playlist generation. In: 2001 IEEE International Conf. on Multimedia and Expo, pp. 329–332, Tokyo (2001)
8. Prim, R.C.: Shortest connection networks and some generalizations. Bell System Technical Journal. 36, 1389–1401 (1957)
9. Pauws, S., Eggen, B.: PATS: Realization and User Evaluation of an Automatic Playlist Generator. In: 3rd Int. Simp. on Music Information Retrieval, Paris (2002)
10. Pauws, S., Verhaegh W., Vossen, M.: Music playlist generation by adapted simulated annealing. Information Science. 178, Issue 3, 647–662 (2008)
11. Eerola, T., Toiviainen, P.:MIDI Toolbox: MATLAB Tools for Music Research, University of Jyväskylä (2004)
12. Gabrielsson, A.: The Relationship between Musical Structure and Perceived Expression. In: Hallan, S., Cross, I., Thaut, M. (eds.) The Oxford Handbook of Music Psychology, pp-1041–150. Oxford University Press (2009)
13. Correa, D., Levada, A. L. M, Costa, L. da F.: Finding Community Structure in Music Genres Networks. In: 12th International Society for Music Information Retrieval Conference, pp. 447–452, Miami (2011)