# Graph-Based Representation of Symbolic Musical Data

**Conference Paper** · May 2009

DOI: 10.1007/978-3-642-02124-4_5 · Source: DBLP

Alexander Hasenfuss
KGS Bad Lauterberg
**30** PUBLICATIONS **507** CITATIONS
SEE PROFILE

Barbara Hammer
Bielefeld University
**432** PUBLICATIONS **6,159** CITATIONS
SEE PROFILE

# Graph-based Representation of Symbolic Musical Data

Bassam Mokbel, Alexander Hasenfuss, and Barbara Hammer

Clausthal University of Technology, Department of Computer Science, Clausthal-Zellerfeld, Germany

**Abstract.** In this work, we present an approach that utilizes a graph-based representation of symbolic musical data in the context of automatic topographic mapping. A novel approach is introduced that represents melodic progressions as graph structures providing a dissimilarity measure which complies with the invariances in the human perception of melodies. That way, music collections can be processed by non-Euclidean variants of Neural Gas or Self-Organizing Maps for clustering, classification, or topographic mapping for visualization. We demonstrate the performance of the technique on several datasets of classical music.

## 1 Introduction

The ever increasing amount of music collections available in online stores or public databases has created a need for user-friendly and powerful interactive tools which allow an intuitive browsing and searching of musical pieces.

Ongoing research in the field of music information retrieval thus includes the adaptation of many standard data mining and retrieval tools to the music domain. In this regard the topographic mapping and visualization of large music compilations combines several important features: data and class structures are arranged in such a way that an inspection of the full dataset as well as an intuitive motion through partial views of the database become possible.

Generally, there are two basic ways to automatically construct the topographic arrangement for a mapping: One way – the classical one – is using a set of features to position each subject in an Euclidean space. Then, since the number of features is usually much larger than three, the high-dimensional spaces have to be projected to two or three dimensions for visualization. Here, usually techniques like the linear Principal Components Analysis (PCA) or the non-linear Self-Organizing Map (SOM) are applied. Unfortunately, it is often not possible to represent complex data such as symbolic musical data, i.e. sequences of notes, by a set of Euclidean vectors. Therefore, there is demand for representations that are capable of capturing complex structures.

A more sophisticated way uses pairwise dissimilarities between all subjects able to capture more complex structures in the data. But in general, these dissimilarities are no longer Euclidean distances and there is no embedding into any Euclidean space without distortion — they may not be metric at all. Hence, the classical methods cannot be applied in this case. Recently, variants of Neural Gas and Self-Organizing Maps for dissimilarity datasets have been introduced

that are able to directly handle arbitrary similarity data instead of only simple Euclidean ones [9].
To make use of those sophisticated non-Euclidean methods, it is essential to have a dissimilarity measure at hand that provides a reliable pairwise measurement of the complex data. For music, there are many different features that can be extracted by algorithmic methods, either from acoustic data or a symbolic description of a musical piece. Global features like the overall tempo, musical key, pitch transition statistics, dynamics statistics or spectral features can be measured for a song and used directly for mapping. However, their importance and level of expression is different in the various styles of music and thus it is difficult to find musical feature sets that are generally valid and equally significant for every genre. A variety of approaches handling musical data based on pairwise comparisons have been proposed, including metrics popular in data mining standards such as the cosine distance based on tf$\times$idf weightings of the basic constituents of given data, complex mathematical constructions such as the Haussdorff distance [17, 6] or the spectra of an associated graph [16].

To compute a dissimilarity between pieces based instead on their tonal and rhythmic progression, it is possible to use the temporal progression of features like rhythmic patterns, note or chord sequences to measure dissimilarity (an overview of encoding methods can be found in [5]). This can be achieved with a suitable method to measure string dissimilarity like the edit distance or the powerful and universal compression distance. Especially the latter has been used in this way on symbolic representations of musical data with promising results in recent years, like e.g. in [2, 4, 13].

Due to the nature of acoustic signals, it usually requires much more effort to extract high-level features from acoustic audio data than from symbolic music representations like MIDI[1] or MusicXML[2]. But recent progress in developing efficient and reliable automated extraction methods that are able to gain musical notation directly from complex acoustic material, as presented in [11, 20, 15], open the way towards mapping techniques which directly rely on a symbolic description of musical data.

When defining a similarity measure on sequences of musical notes, certain invariances should be respected to comply with the average human perception of melodies: These include invariances to transposition of the notes to a different key and the scaling of the tempo (further described in [7]).
In the following section, we introduce a new way to convert symbolic representations of music into strings via priorly constructed precedence trees. We show how the string encoding derived from the graph structure as well as the subsequently applied Normalized Compression Distance (NCD) is beneficial for the named invariances.
We implemented our method in Matlab[3] and used MIDI files as symbolic input data. We processed selected subsets of classical pieces from the comprehensive *Kunst der Fuge*[4] MIDI collection, containing pieces from almost a thousand com-

[1] `http://www.midi.org`
[2] `http://www.musicxml.org`
[3] `http://www.mathworks.com`
[4] `http://www.kunstderfuge.com`

posers, spanning various musical forms and epochs. The generated dissimilarities were mapped with a non-metric Multi-Dimensional Scaling with Kruskal's normalized stress1 criterion. The experiments show most of the data arranged in meaningful clusters with a reasonable separation of composers and eras.

## 2 Graph-based Representation of Symbolic Musical Data

To measure the dissimilarity between the tonal and rhythmic progression of two pieces of music, our method compares string representations derived from their note succession. We therefore developed an algorithm that converts the symbolic note sequences in a MIDI file into a string, following a priorly constructed precedence structure. Our algorithmic approach is based on the assumption that a human's subjective perception of musical identity usually works very context-driven. Thereby we suppose that most listeners will consider a melody to a certain extent similar to a copy of it, if it has been changed in the following ways:

- It is shifted in its overall pitch, i.e. it has been transposed to another fundamental note.
- It is scaled in its overall tempo, i.e. all note lengths and pauses have been contracted or elongated by a constant factor.

Thus we assume that the human perception of melodies is, to a certain extent, invariant to overall pitch translation (*pitch translation invariance*[5]) and to an overall scaling of the tempo (*time scaling invariance*). To gain a measure that is close to the human music perception we therefore encode the note sequences to new symbolic sequences with an encoding method that is invariant to the aforementioned changes of the note sequences. That means it produces the same output, whether the input is the original or the altered note sequence. The information that is not encoded in the new strings is the magnitude by which the pitch was shifted or the tempo scaled. As the described human assessment of similarity would probably decrease along with a raise in magnitude of such changes, it might be more truthfully described by distinguishing degrees of similarity. Although this is not part of our encoding scheme at the moment, it is easily imaginable to incorporate such information into our measure in the future.

Some related methods can be found in literature that partially provide for the emphasized invariances, like e.g. [18, 19, 4, 13]. In [4] pitch translation invariance was achieved with a global pitch normalization throughout the entire piece, making the encoding very sensitive regarding the automatic choice of the global point of reference. In [13] every note's pitch was encoded as the difference to the pitch of its directly preceding note. In addition to independence of the overall pitch, this method yields local separation: parts in the strings are equal for parts of two songs that, aside from transposition, have equal note sequences, even if the rests of these songs are completely different. Using common string

[5] also referred to as *pitch invariance or transposition invariance.* The terms differ in literature, we use the ones described in [7]

dissimilarity measures on those two representations would therefore reflect the partly equality in its output value. In addition, one could store note lengths and pauses analogously to gain time scaling invariance. Still, in these strings you will find only very little equality in the case of two songs playing the same melody (like a riff or a theme), only with dissimilar accompanying notes. Then the output of a string dissimilarity measure would in our opinion not represent most listeners' assessments.

Our goal for the generated string representation was therefore a decomposition of the tonal and rhythmic progression of a song, that has the benefit of local points of reference, but, on top of that, represents melodic lines and themes more independently of the surrounding melodic context. Our strategy is to automatically define precedence relationships between notes throughout the entire piece: The functions $pitch(n)$, $start(n)$ and $length(n)$ return the absolute pitch and the normalized start time and length of a note $n$ respectively. For every note $n$ in the sequence of notes $N$ played in the song, the algorithm picks one designated predecessor. For the current note $cn \in N$, the function $pred(cn)$ returns one of its time-wise preceding notes on the same MIDI channel (the sequence $P(cn)$) as the predecessor note.

We define

$$pred(cn) = \underset{p \in P(cn)}{\operatorname{argmin}} \left( \; k \cdot |pitch(p) - pitch(cn)| + \; r \cdot \frac{start(cn) - start(p)}{length(cn)} \; \right)$$

with $P(n) = \{ \, x \in N \; : \; start(x) < start(n) \, \}$ and $p \in P(cn)$, $cn \in N$.

The predecessor is thereby chosen to be the closest prior note in terms of the absolute difference in start time *and* in pitch. Since the time-wise distance is calculated relatively to the current note's length, the field of search is stretched or shrinked in the time dimension depending on its length. As an alternative strategy one could also consider stop times of prior notes instead of their start times. The global parameters $k$ and $r$ determine the overall search strategy.

From a graph-theoretical point of view the resulting precedence structures form a forest of weighted trees whose vertices represent the played notes. Examples are shown in Figure 1 and 4. The trees are then utilized to store the change in pitch and length as well as the relative difference in note start times at every edge along every path of every tree. For a note $n$ the named changes are calculated and stored in relation to its predecessor $pr = pred(n)$ as follows:

$$relpitch(n) = pitch(n) - pitch(pr),$$
$$reltiming(n) = \frac{start(n) - start(pr)}{length(pr)}, \;\; rellength(n) = \frac{length(pr)}{length(n)}.$$

Next to $rellength(n)$, the value of $reltiming(n)$ adds some more information about the rhythmic expression to the string representation, as it also encodes the existence and length of pauses or overlaps between the notes. Since the start times and note durations are being normalized beforehand, the results from $rellength(n)$ and $reltiming(n)$ stay the same for varying musical tempo

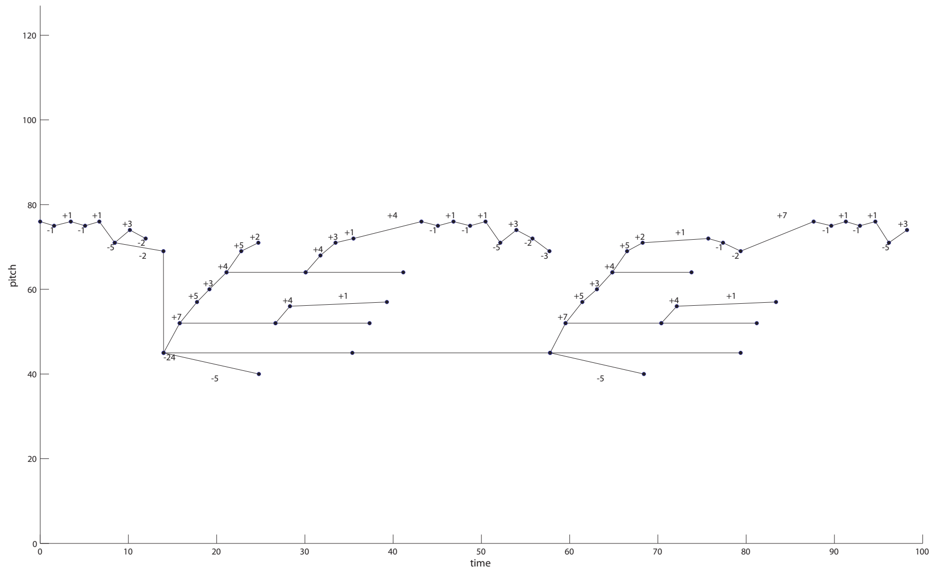

**Fig. 1.** The precedence tree structure for the first 60 notes of Beethoven's "Für Elise". The edges are marked with all nonzero pitch changes of notes relative to their predecessors.

notations that represent the same musical output with different combinations of overall tempo and note durations.

Considering the precedence structure, a small, local change in the musical progression will subsequently cause it to alter locally, resulting yet only in a local symbolic change of the string representation. To explain the benefit of this behavior on a practical example, imagine a bass line which is - due to its lower pitch - isolated in the tonal progression. Its melody would be represented independently in the string, unaffected by changes of higher lead melodies played at the same time on the same MIDI channel. In this way any two melodies will have independent representations within the string as long as their note sequences are sufficiently separated in pitch locally.

To sum it up, our encoding method is fully *pitch translation invariant* and *time scaling invariant* on an entire song but also shows highly invariant behavior upon changes to certain subsets of notes and hence to variations of melodies. It offers a structural decomposition for the representation of polyphonic music or polyphonic instrument tracks.

To calculate the dissimilarity of the string representations, we used the popular Normalized Compression Distance (NCD) (see [3]), a measure based on approximations of the Kolmogorov Complexity from algorithmic information theory described in [12]. The NCD is defined as

$$\mathrm{NCD(x,y)} = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

where x and y are strings, C(x) denotes the compressed size of x and C(xy) the compressed size of the concatenation of x and y using a real compressor. For our experiments the bzip2 compression method was used. Since bzip2 works byte-oriented as most of the common compression methods, the size of a reasonable set of symbols is restricted to $2^8$. Therefore, we utilize the integer values in [1..255] to code every possible relative state change of a note compared to its predecessor. For every musical piece, we automatically build two strings, one that holds the pitch changes $relpitch(n)$ for every note $n$ and another one for the rhythmic progression. The latter is compiled from $rellength(n)$ and $reltiming(n)$, resulting in a string which is twice as long as the one representing the pitches. The dissimilarity of two songs is then calculated as the mean value of the normalized NCD of the pitch strings and the normalized NCD of the rhythmic strings. If one disregards the possibility to change the overall tuning of a MIDI file or the use of pitch-bending and vibrato while notes are played, MIDI distinguishes at most 128 different note pitches [0..127]. So our byte representation has to cover a total of 255 possible values of relative pitch change, 127 upwards and 127 downwards plus one for 'no change' which in total is presentable with one byte. Our code thus indicates decreasing pitches with the values [1..127], 'no change' with 128 and increases with [129..255]. To encode the fractions of note durations given by *rellength*, our algorithm maps all occurring numeric values to 127 real-valued intervals that are centered around rhythmically important ratio values. These are all the possible fractions between the lengths of a whole note, a half, a quarter, an eighth, a $\frac{1}{16}$th, a $\frac{1}{32}$th and a $\frac{1}{64}$th note as well as the corresponding triplets and five-lets in between. The encoding thereby treats a $\frac{1}{64}$th note followed by a whole as the farthest upward change in note length and downward vice versa. This way, very steep transitions of note lengths which exceed this magnitude are all being treated as equal maximal steps as they are assigned to the same interval. The resulting unique ratios consist of 63 values that are less than 1 (meaning the duration of the actual note is longer than the predecessor's), another 63 larger than 1 (the actual note is shorter than the predecessor), and the ratio equal to 1 (durations are equal). These 127 ratios are presentable with half a byte. The other byte values are used analogously as symbols to encode the values of *reltiming*.

Apart from information about the instrumentation, our coding scheme disregards musical phrasings like pitch-bending, vibrato, etc. We are currently working on an algorithm to correctly convert seamless pitch transitions (i.e. bendings) to discrete notes. After the conversion our algorithm would include the emulated pitch-bending like normal notes. This opens the way to further test our dissimilarity measure with datasets of mainstream and popular music. In popular music, phrasings are usually very important in the melodic progression, especially in the notations of the vocal/singing voices.

## 3 Experiments

We implemented our algorithms in Matlab 7.5 and used the Matlab MIDI Toolbox [8] to read the MIDI files. To show the performance of the introduced dissim-

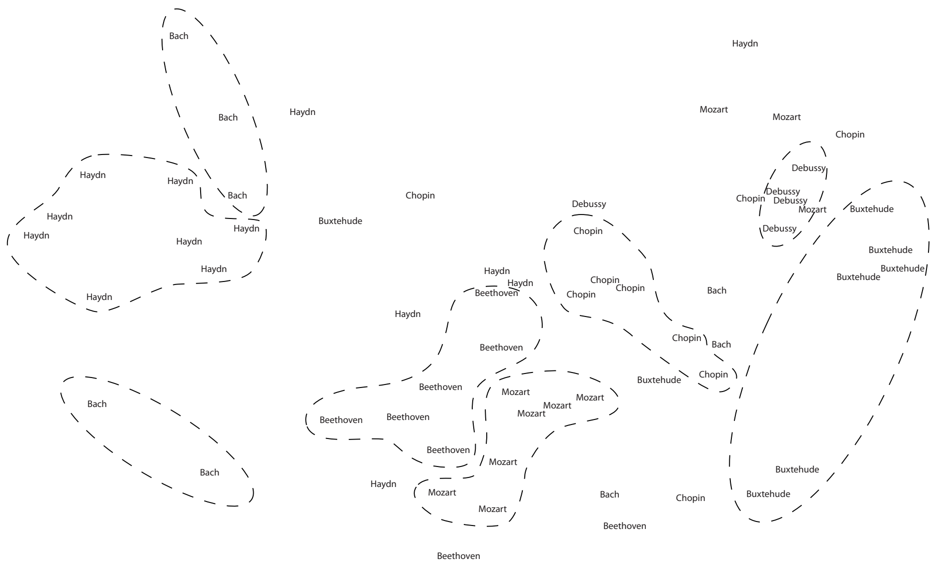
**Fig. 2.** Performance of the Dissimilarity Measure on a Classical Piano Dataset

ilarity measure, we used Multi-Dimensional Scaling to embed the dissimilarities into the Euclidean plane for visualization.

First, we measured a dataset taken from [4] and projected the calculated dissimilarities into the Euclidean plane by non-metric Multi-Dimensional Scaling with Kruskal's normalized stress1 criterion. The dataset consists of 63 classical pieces of piano music originating from seven composers (Bach, Beethoven, Buxtehude, Chopin, Debussy, Haydn, Mozart). As can be seen in Figure 2 most of the data is arranged in meaningful clusters. Obviously, the dissimilarity measure distinguishes between the different composers, though it is solely based on the dissimilarities in tonal and rhythmic progressions.

The second experiment shows a dissimilarity-based mapping of symphonies from the classical period. Symphonies by Beethoven, Haydn, and Mozart were measured by the above introduced method. The symphonies have been separated into movements beforehand such that each of the 190 data points corresponds to a movement of a symphony. The mapping of the data is shown in Figure 3. The three clusters are clearly recognizable, two clusters are well separated and the third one (Haydn) is in between. This very promising result demonstrates the abilities of the new graph-based measure since it is consistent with the music historical setting.

## 4 Summary and Outlook

In this work, we have introduced a novel approach for measuring the similarity of symbolic music. The presented approach features a graph-based representation and derives a dissimilarity measure thereof. These dissimilarities can directly be

processed by topographic mapping techniques like e.g. Relational Self-Organizing Map or Relational Neural Gas (cf. [10]). The used graph precedence structure is built on pitch transitions and relative timings. We demonstrated the performance of the measure on several datasets from classical music.
In the near future we want to further test our method on very large datasets of different musical styles, especially with popular and mainstream music. To compare its performance and speed with existing approaches, we plan to use test datasets for music classification.

Furthermore, we want to implement alternative methods to measure the similarity of the strings derived from the graphs other than the compression metric. Moreover, we are currently working on the application of graph similarity measures with graph kernels as presented in [14].
By combining our system with a prior automatic conversion of audio data to note sequences, as presented in [11, 20, 15], many further applications are possible. One obvious extension would be a 'query by humming' database search system, as e.g. in [1]. By preprocessing the entire dataset with a topographic mapping method for non-Euclidean data like the Relational Self-Organizing Map, an effective similarity-based search technique is possible that narrows down the search space.

To speed up the calculation of the dissimilarity measure, it might be sufficient to calculate the dissimilarities only between the most-significant paths — the paths with the highest entropy in pitch variations. Those are most likely to be the melodic progressions that have a lead role within the arrangement.

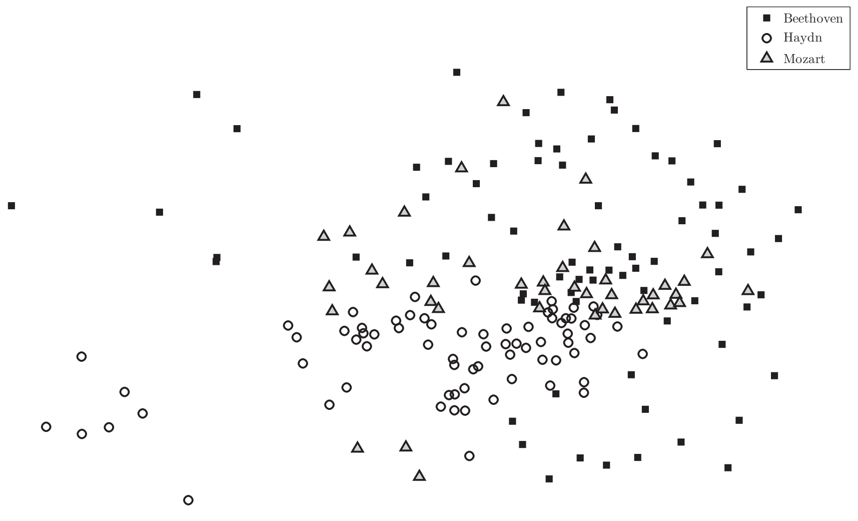

**Fig. 3.** Dissimilarity-based Mapping of Symphonies of Beethoven, Haydn, and Mozart – The Symphonies have been separated into movements

## References

1. W. Birmingham, R. Dannenberg, and B. Pardo. Query by humming with the vocalsearch system. *Commun. ACM*, 49(8):49–52, 2006.
2. Z. Cataltepe, Y. Yaslan, and A. Sonmez. Music genre classification using midi and audio features. *EURASIP Journal on Advances in Signal Processing*, page Article ID 36409, 2007.
3. R. Cilibrasi and P. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, April 2005.
4. R. Cilibrasi, P. Vitányi, and R. de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.
5. P. P. Cruz-Alcázar and E. Vidal. Two grammatical inference applications in music processing. *Applied Artificial Intelligence*, 22(1&2):53–76, 2008.
6. P. Di Lorenzo and G. Di Maio. The hausdorff metric in the melody space: A new approach to melodic similarity. In *ICMPC*, 2006.
7. P. Dorrell. *What Is Music?: Solving a Scientific Mystery.* Lulu (Print on Demand), 2005.
8. T. Eerola and P. Toiviainen. *MIDI Toolbox: MATLAB Tools for Music Research.* University of Jyvaskyla, 2004.
9. B. Hammer and A. Hasenfuss. Relational neural gas. In J. Hertzberg, M. Beetz, and R. Englert, editors, *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 190–204, Berlin, 2007. Springer.
10. B. Hammer and A. Hasenfuss. Relational neural gas. In J. Hertzberg, M. Beetz, and R. Englert, editors, *KI 2007: Advances in Artificial Intelligence,30th Annual German Conference on AI, KI 2007*, volume 4667 of *Lecture Notes in Artificial Intelligence*, pages 190–204, Berlin, 2007. Springer.
11. A. Klapuri and M. Davy, editors. *Signal Processing Methods for Music Transcription.* Springer, New York, 2006.
12. M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer, 1997.
13. A. Londei, V. Loreto, and M. O. Belardinelli. Musical style and authorship categorization by informative compressors. In *Proc. ESCOM Conference*, 2003.
14. M. Neuhaus and H. Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines.* World Scientific, 2007.
15. B. Pardo and W. P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.
16. A. Pinto, R. H. van Leuken, M. F. Demirci, F. Wiering, and R. C. Veltkamp. Indexing music collection through graph spectra. In *Proceedings of the International Conference of Music Information Retrieval*, 2007.
17. C. A. Romming and E. Selfridge-Field. Algorithms for polyphonic music retrieval: The hausdorff metric and geometric hashing. In *Proceedings of the International Conference of Music Information Retrieval*, 2007.
18. A. Ruppin and H. Yeshurun. Midi music genre classification by invariant features. In *Proceedings of the International Conference of Music Information Retrieval*, 2006.
19. E. Ukkonen, K. Lemström, and V. Maekinen. Geometric algorithms for transposition invariant content-based music retrival. In *Proceedings of the International Conference of Music Information Retrieval*, 2003.
20. J. Woodruff and B. Pardo. Using pitch, amplitude modulation, and spatial cues for separation of harmonic instruments from stereo music recordings. *EURASIP Journal on Advances in Signal Processing*, page Article ID 86369, 2007.

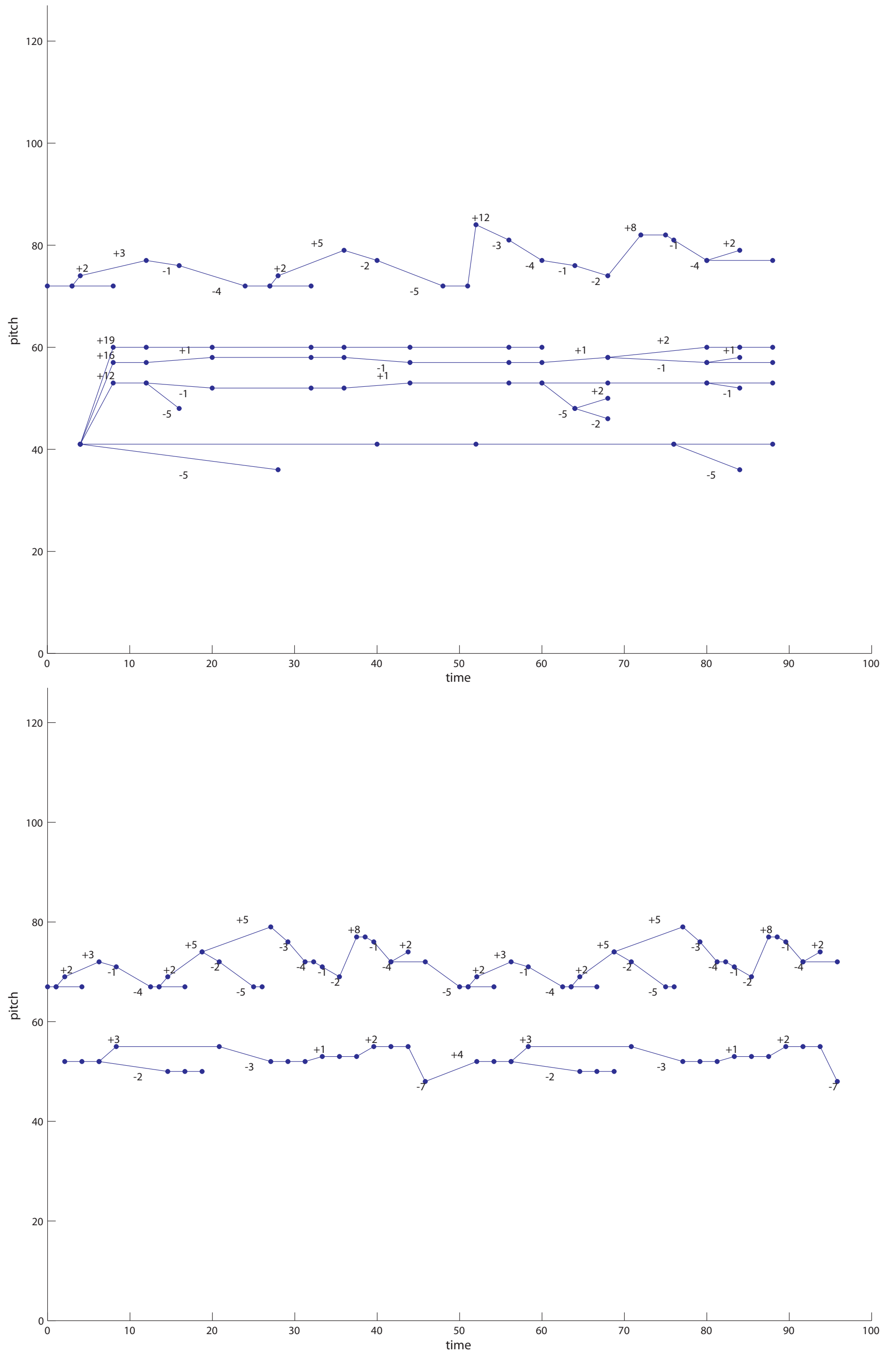

**Fig. 4.** Graph-based Representations of the Song *Happy Birthday* – Similar Lead Melodies with Different Accompaniments