Automatic Reduction of MIDI Files Preserving Relevant Musical Content

Søren Tjagvad Madsen[1,2], Rainer Typke[2], and Gerhard Widmer[1,2]

[1] Department of Computational Perception, Johannes Kepler University, Linz
[2] Austrian Research Institute for Artificial Intelligence (OFAI), Vienna

Abstract. Retrieving music from large digital databases is a demanding computational task. The cost for indexing and searching depends not only on the computational effort of measuring musical similarity, but also heavily on the number and sizes of files in the database.
One way to speed up music retrieval is to reduce the search space by removing redundant and uninteresting material in the database. We propose a simple measure of 'interestingness' based on music complexity, and present a reduction algorithm for MIDI files based on this measure. It is evaluated by comparing reduction ratios and the correctness of retrieval results for a query by humming task before and after applying the reduction.

1 Introduction

Retrieving music from large digital databases is a demanding computational task. As an alternative to the brute force approach of searching everything in the data base, we propose to preprocess each item in the data base by removing uninteresting material in an intelligent way inspired by results from music cognition. As music collections can be very large, simplifying the data will allow us to search more files within the time/space limitations.

We introduce a number of reduction mechanisms and evaluate them by comparing storage requirements and the correctness of retrieval results for a music query task before and after applying the reduction. Initial results indicate that the approach provides a good tradeoff between the correctness of the retrieved files and the amount of additional data that can potentially be searched.

The paper is organised as follows. Section 2 describes the music search engine. Section 3 introduces the conceptual background of the reduction algorithm which is presented in section 4. Finally we report our findings in section 5.

2 A Melody Search Engine

We test our reduction method with the aid of a melody search engine. The MIDI search engine of Musipedia.org was applied to a set of 1000 MIDI files that were also used for the "symbolic melodic similarity" task of MIREX 2006[3]. For this

[3] The Music Information Retrieval Evaluation eXchange,
`http://www.music-ir.org/mirex/2006/index.php/Main_Page`

set, some ground truth is known, and it is the hope that search engine will be able to retrieve the same files when querying the unprocessed and the reduced files. The web-based Musipedia searches about 100,000 mostly polyphonic MIDI files for melodic user queries.

2.1 The Search Task

Given a short user query (about 5 to 25 notes), the search engine will return the MIDI files that contain a sequence of notes which are melodically similar, that is, form a similar pattern of onset times and pitches. This is done in a transposition-invariant and tempo-invariant way. Results are ordered by similarity of the matching melody according to the Earth Mover's Distance [13]. Before queries can be executed, the search engine builds an index of all material in the database. This computationally expensive task is performed only once, and the index can then rapidly be queried.

2.2 Extracting Monophonic Melodies from Polyphonic MIDI Files

As a preparation for finding melodies, the polyphonic MIDI files are first split into monophonic voices by using a skyline algorithm for every channel or track. That is, MIDI files with multiple tracks are split into single-track MIDI files, and MIDI files with multiple channels into single channel MIDI files. For every single-track or single-channel MIDI file, a skyline algorithm inspired by [14] and [1] sorts the notes first by pitch (descending) and then by onset time (ascending). Then, starting with the highest pitch, the note with the earliest onset time – let us call it n – is marked as "not to be deleted". Every note with an equal or lower pitch and an onset time that is either the same or lies within 60 % of the duration of note n after n's onset is then deleted. From n, the algorithm proceeds to the next still existing and still unmarked note with the same pitch and treats it in the same way as n. Once all notes with n's pitch have been processed, the algorithm continues one half-tone lower, or it terminates in case there is no other unmarked note left. In the end, the remaining notes will form a sequence of the highest pitched notes in the voice.

Obtaining monophonic representations of polyphonic tracks and channels is essential for being able to measure the musical similarity. Music similarity for polyphony is much harder to define than music similarity between sequences.

2.3 Problem: Identifying Melodic Musical Material

The musically rather uninformed skyline algorithm turns out to be surprisingly good at agreeing with humans about what the melodic line is in a polyphonic set of notes [14]. However, it does not tell us whether there is anything melodic at all about a given sequence of notes. The whole track or channel on which the algorithm operates might contain only accompaniment figures which are very unlikely to ever be searched for by human users who would like to identify a

melody or retrieve a MIDI file containing a given melody. Adding such potentially uninteresting material to the search engine's index makes the index unnecessarily large and slows down the search process.

If one could identify those notes which human users will probably search for, and then only index those notes, one should be able to work with a smaller index and potentially me able to include more files within the same space. But the melody search engine should still be able to give almost the same search results, so a conservetive approach would be preferred. In the next sections, we describe and evaluate an algorithm which does exactly this – it reduces the size of MIDI files by excluding musical material which is unlikely to be perceived as relevant by humans.

3 Music Complexity as a Measure of Interestingness

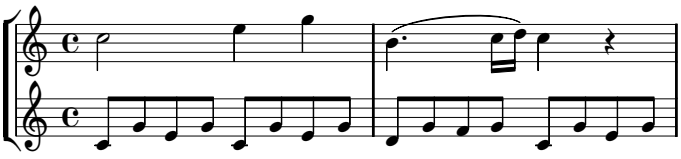
The basic motivation for our model of melody identification is the observation, which has been made many times in the literature on music cognition, that there seems to be a connection between the complexity of a musical line, and the amount of attention that will be devoted to it on the part of a listener [12]. A voice introducing new or surprising musical material will potentially attract the listener's attention. This effect will, however, decrease if the new material is constantly repeated. We will pay less and less attention to it and become habituated or accustomed to the stimulus.

We postulate that in many situations this phenomenon of interestingness can be modeled by quantifying the amount of information that a given note sequence contains. If we assume that the musical foreground is the musical material that presents most new information and commands most attention, it seems natural to investigate music complexity as a method for detecting such material.

The idea of using information-theoretic complexity measures to characterise aspects of musical development is not at all new. After the advances in information theory in the fourties, Leonard Meyer [9] was one of the first to relate music to information dynamics. He attempted to explain emotions and meaning in music in by examining the communication of uncertainties and expectations in music. Complexity has also been shown to be related to liking [10] in such a way that we tend to like the music the most that have medium complexity – too simple and too complex music is less preferred. Also, music complexity has been used for clustering music in genres [2, 11, 5], and in [4], a measure of Information Rate computed over a piece of music was shown to correlate in significant ways with familiarity ratings and emotional force response profiles by human subjects.

3.1 Measuring Musical Complexity

Earlier experiments of ours have shown that entropy of note events can with some success be used to point out melody notes and melody tracks in MIDI files [8, 7]. Shannon's entropy is a measure of randomness or uncertainty in a signal. If the predictability is high, the entropy is low, and vice versa. Let X be



	H_C	H_D	$H_{C,D}$
Top voice	2.128	1.842	2.807
Lower voice	1.906	0.0	1.906

Fig. 1. Measuring entropy of the two voices in the first two measures of the first movement of Mozarts piano sonata KV 545.

a discrete random variable on a finite set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ with probability distribution $p(x) = Pr(X = x)$. Then the entropy $H(X)$ of X is defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x). \tag{1}$$

X could for example be the set of MIDI pitch numbers and $p(x)$ would then be the probability (estimated by the frequency) of a certain pitch. In the case that only one type of event (one pitch) is present in the window, that event is highly predictable or not surprising at all, and the entropy is 0. Entropy is maximised when the probability distribution over the present events is uniform.

We are going to calculate entropy of 'features' extracted from the notes in monophonic lines. We will use features related to pitch and duration of the notes. A lot of features are possible: MIDI pitch number, MIDI interval, pitch contour, pitch class, note duration, inter onset interval etc. (see for example [3]). We will focus on the following two features in the model presented here:

1. Pitch class (C): the occurrences of different *pitch classes* present (note name, irrespective of the octave; MIDI pitch modulo 12).
2. Note duration (D): the note inter onset interval (IOI) classes present, where classes are derived by discretisation (an IOI is given its own class if it is not within 10% of an existing class).

We can measure the entropy of a given feature (H_C, H_D) in a sequence of notes by extracting note events, and calculate entropy from the frequencies of these events. As the features are meant to be calculated from successive single musical events (rather than arbitrary polyphonic structures), we will apply these measures only to skylined tracks or channels as in the case of the melodic search described in section 2.2.

Entropy is also defined for a pair of random variables with joint distribution, allowing us to consider the joint events of a certain pitch and a certain duration. We will denote the joint entropy measure of pitch class and IOI class with $H_{C,D}$.

As an example of applying the complexity measures to music, consider Figure 1. The three complexity measures just described were applied in turn the two

musical lines found in the first two measures of a Mozart piano sonata. All three complexity measures show that the top voice contains the most information. In this case, most listeners would also agree that the upper voice is indeed the most interesting one, so this example supports our hypothesis that the most complex voice is experienced as foreground.

As another intuitive example, consider four bars of pop music. Typically a solo instrument (voice, saxophone) will be playing a melody or a theme on top of a number of accompaniment instruments (rhythm guitar, drums). While the rhythm section will typically play short repeated pattern (e. g. 1–2 bars), the melody line often performs longer passages before it repeats itself (e. g. all four bars). When measuring information with entropy within the four bars, the voice playing repeated patterns doesn't add any new information for each repeat – the entropy measured on all repetitions is equal to the entropy of the pattern itself as the distribution of events is the same. So the instrument playing longer phrases is likely to be considered more complex. In this sense the entropy measure is capable of taking habituation into account.

4 Reducing a MIDI File

The reduction algorithm is straightforward. First each voice (recognised by MIDI track and MIDI channel) is skylined in order to get monophonic sequences. All notes that do not make it through this process are marked. Then a window is slid over the notes in small steps. From the notes not marked by the skyline algorithm in each voice segment in each window, an interestingness value is calculated, and stored in the window. Then a threshold for including or excluding notes is introduced. We only want to keep the notes occurring in the most interesting e. g. 50 % of the segments in the windows. By sorting all interestingness values, we obtain the actual threshold value by picking the value positioned at the wanted delimiting point (e. g. the median value).

Again, each voice segment in each window is considered. If the interestingness value of the note sequence is above the threshold, these notes are marked with the 'write' label. In the end, all notes that occur in any voice segment having an interestingness value higher than the limit will be given the write label.

The reduced MIDI file can now be written. Only notes marked with the 'write' label are written. For each file, we report the ratio of notes that were deleted by skylining and the ratio of notes that were removed by the complexity measure.

In section 5 we report how this basic algorithm fares when evaluated in a Query-by-Humming (QBH) context. We also examined one variant of the algorithm. Instead of using just one measure of interestingness, we can in turn mark notes using different interestingness measures – e. g. mark notes with a rhythmic measure as well as a pitch based measure, and write notes that have been marked by any of the measures. This approach will then mark notes that are interesting either because of rhythm or pitch.

Query	Ground truth
q000001	(k001051,k000061,k000258)
q000002	(k000589),(k004517,k092387,k096238)
q000003	(k000589),(k004517,k093757)
q000004	(k095188,k095520)
q000005	(k094777),(k096523)

Table 1. The ground truth for the MIREX 2006 "Karaoke" collection created by running 10 algorithms, having human graders assign similarity scores, and averaging and sorting the retrieved items by rough score. Items with equal average rough scores are grouped together, here shown with parentheses.

5 Experiments

For a music search engine, the reduction of MIDI files can be an advantage, if the database size can be reduced noticeably while not losing any true positives for typical queries. At least the number of lost true positives should be as small as possible – any reduction of the number of true positives should be smaller than the reduction of the database size.

To evaluate how our reduction method performs, we used a collection of 1000 polyphonic MIDI files from the 2006 MIREX symbolic melodic similarity task. For this collection, a ground truth of matching songs is known for a set of five queries. Our music search engine was the best performing algorithm at MIREX 2006, and we want to compare the songs returned by the algorithm before and after reducing the data.

The ground truth was established as part of MIREX 2006 as follows: All ten participating algorithms were used to retrieve the ten most similar items for every query. This resulted in a pool of up to 100 retrieved items per query. Human graders then manually compared the queries to the retrieval results and assigned similarity scores to each query-retrieval result pair. Both a rough score (1 = very similar; 2 = somewhat similar; 3 = not similar) and a fine score (a number in the range from 1 to 10) were collected from the graders. An overall ground truth was then established by calculating the average rough score for every retrieved item, throwing away items with an average score worse than 2 ("somewhat similar"), and finally grouping those items together whose average rough score was the same. This resulted in an ordered ranked list, where some items have equal ranks (see Table 1).

Query q000002 and q000003 refer to the same song, but q000003 is a more challenging, or 'badly sung' query. This explains the large overlap in the ground truth: the human judges agree that for both queries, the MIDI file k000589 contains a melody which is very similar to the query. However, only the items which are found in the ground truth for both queries, k000589 and k004517, really contain the query melody (ABBA's 'The winner takes it all'), and these files are almost identical arrangements of the song. Items k092387, k092387, and k096238 refer to other and quite different songs. We do not see the similarity the

human judges claimed to see, and therefore we believe that one should take the ground truth for Queries q000002 and q000003 with a grain of salt. The ground truth for the other queries is more believable since for each of the remaining queries, all items in the ground truth are cover versions of the same piece.

In Table 2, we show the ranked lists returned by the retrieval algorithm with the original data and with four different reductions of the data.

The first reduction used the joint pitch class and IOI entropy measure $H_{I,D}$. The second reduction used a linear combination of $H_{I,D}$ and H_D – giving a little more weight to rhythmic differences. In the third experiment, three measures were used in turn to mark the interesting notes as explained in section 4. The threshold was set to 50 % or 60 % (discarding interestingness values under this limit), and in these three cases the window size and hop size was 4 seconds and 1 second respectively. We notice that the different parameters result in different reduction rates. The reduction rate is the percentage of notes that was removed by the interestingness measure (after skylining) averaged over all files.

The queries q000001, q000004, and q000005 seem to retrieve the same files in all data sets (except in the highly reduced reduction 2). Queries q000002, and q000003 (that refer to the same song) result in some changes in the ranking delivered by the search engine, and some of the ground truth songs are ranked very low.

We did a fourth experiment, using the same complexity measure as in Reduction 2, but using a longer (10 sec) window (and a threshold of 0.5). This eliminated 35.34 % of the notes on average over all files in the Karaoke data set. For four of the queries, the same songs could be retrieved from the unprocessed and the summarized dataset. For the remaining query (q000003), the ground truth files were not retrieved from the reduced dataset, but when listening to the reduced files, they seem to contain all instances of the melody that the query refer to, so the hit in the unreduced song might have been due to a match of some unintended part of the song. After all the query is a poor interpretation of the ABBA song.

Also for this reduction, nine of the ten songs retrieved for query q000002 on the unreduced data appear among the 12 first matches for the same query in this fourth reduction. This tells us that indeed many of the important and interesting passages in the music files are preserved.

When listening to reduced files, it becomes apparent that the original arrangements are highly recognizable. Most of the important material is still there - mainly accompaniment figures and drum loops have been removed.

Figure 2 shows on the left five bars from one of the ground truth files (k000589) in the Karaoke dataset (transcribed and quantised by the authors). On the right the same passage is depicted after applying the skylining and reduction using the settings from our fourth experiment. The top staff containing the usually sung melody is kept intact. The second voice (next two staffs) holds a piano accompaniment that was completely omitted, probably due to the fact that it was rhythmically uninteresting as there is an onset on every eighth note.

Query	**Before reducing**	**Reduction 1**	**Reduction 2**	**Reduction 3**	**Reduction 4**
q000001	*k000258,* *k000061,* *k001051*	*k000258,* *k000061,* *k001051*	*k000061,* *k000258,* *k001051*	*k000258,* *k000061,* *k001051*	*k000258,* *k000061,* *k001051*
q000002	*k004517,* *k000589,* k000076, k002413, k000034, k092493, *k092387,* k092386, k000843, *k096238*	*k004517,* *k000589,* k000836, k092948, k000076, k000034, k000861, k092493, *k092387,* k092386, k000843, *k096238*	k000836, k092948, k000076, k000916, k001012, k000034, k000861, *k092387,* k092386, k000843, *k096238,* ...	*k004517,* *k000589,* k000836, k092948, k000076, k000034, k000861, k092493, *k092387,* k092386, k000843, *k096238*	*k004517,* *k000589,* k000836, k092948, k000076, k000034, k000861, k092493, *k092387,* k092386, k000843, *k096238*
q000003	*k004517,* *k000589,* k000780, ...	k000780, k000899, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, k000234, k092442, ...	k000780, k000899, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, *k000589,* *k004517,* ...	k000780, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, *k000589,* *k004517,* ...	k000780, k000899, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, k000234, k092442, ...
q000004	*k095188,* *k095520*	*k095188,* *k095520*	*k095520,* k000811, ...	*k095188,* *k095520*	*k095188,* *k095520*
q000005	*k094777,* *k096523*	*k094777,* *k096523*	*k094777,* *k096523*	*k094777,* *k096523*	*k094777,* *k096523*
Reduction		27.15 %	36.71 %	15.50 %	35.34 %
Measure		$H_{C,D}$	$\frac{2}{3}(H_{C,D}) + \frac{1}{3}H_D$	H_C & H_D & $H_{C,D}$	$\frac{2}{3}(H_{C,D}) + \frac{1}{3}H_D$
Window size		4 sec	4 sec	4 sec	10 sec
Hop size		1 sec	1 sec	1 sec	0.8 sec
Threshold		0.5	0.6	0.5	0.5

Table 2. Retrieval results for the original data and for three different reductions. True positives are printed in *italics.*

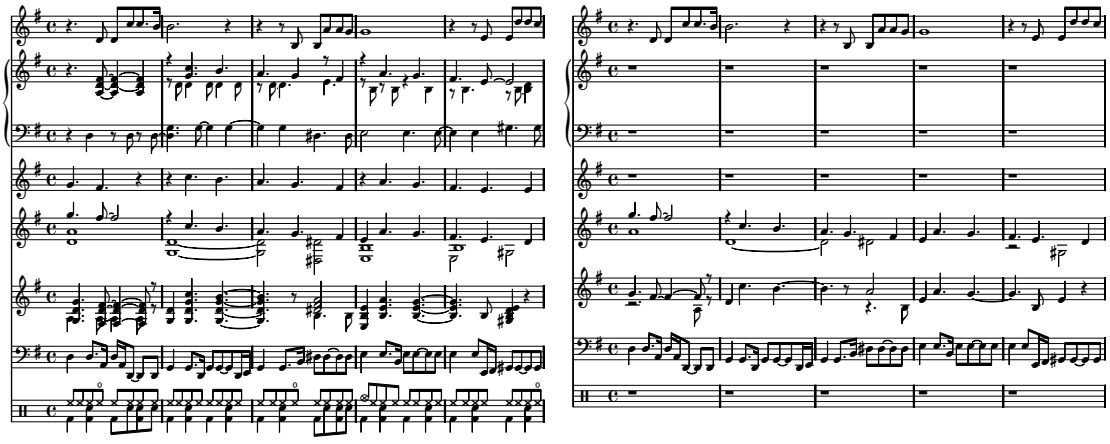


Fig. 2. Five measures of an ABBA song, the original on the left and the summary on the right.

The highly active bass line (second last staff) was kept, and the very repetitive drum track was removed.

6 Conclusion and Future Work

We presented an approach to identifying interesting melodic lines in MIDI files. A MIDI file reduction algorithm based on this approach was presented and evaluated on a QBH task. The current results with a limited evaluation scenario suggest that our approach is a promising path to follow. Future work include testing the approach on more data.

The reduction algorithm was explicitly designed for the QBH task, optimised to preserve interesting melodic lines in MIDI files. Although we (immodestly) find that the summarised files sound like good representations of the originals, it would be interesting to explicitly seek to conserve the *structure* of the composition e. g. in terms of harmonic content (for example, in the current approach it is not guaranteed that the bass line will be kept). This could be useful when summarising MIDI files to be played on hardware that can only play a limited number of notes simultaneously – e. g. to automatically make a standard MIDI file comply with the Scalable Polyphony MIDI file format [6]. This is a different reduction goal where we also think the measures of interestingness can play an important role.

7 Acknowledgments

This research was supported by the Viennese Science and Technology Fund (WWTF, project CI010), the Austrian Research Fund (FWF), Project number

M1027-N15 and project P19349-N15, and by the Austrian Federal Ministries of Education, Science and Culture and of Transport, Innovation and Technology.

References

1. Wei Chai. Melody retrieval on the web. Master's thesis, MIT, 2001.
2. Rudi Cilibrasi, Paul Vitányi, and Ronald de Wolf. Algorithmic clustering of music based on string compression. *Computer Music Journal*, 28(4):49–67, 2004.
3. Darrell Conklin. Melodic analysis with segment classes. *Machine Learning*, 65(2-3):349–360, 2006.
4. S. Dubnov, S.McAdams, and R. Reynolds. Structural and affective aspects of music from statistical audio signal analysis. *Journal of the American Society for Information Science and Technology*, 57(11):1526–1536, September 2006.
5. Ming Li and Ronan Sleep. Genre classification via an LZ78-based string kernel. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, London, U.K., 2005.
6. Simon Lui, Andrew Horner, and Lydia Ayers. MIDI to SP-MIDI transcoding using phrase stealing. *Multimedia, IEEE*, 13(2):52–59, 2006.
7. Søren Tjagvad Madsen and Gerhard Widmer. A complexity-based approach to melody track identification in midi files. In *Proceedings of the International Workshop on Artificial Intelligence and Music (MUSIC-AI 2007) held at the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, Hyderabad, India, January 2007.
8. Søren Tjagvad Madsen and Gerhard Widmer. Towards a computational model of melody identification in polyphonic music. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, Hyderabad, India, January 2007.
9. Leonard B. Meyer. *Emotion and Meaning in Music*. The University of Chicago Press, Chicago, 1956.
10. Mark G. Orr and Stellan Ohlsson. Relationship Between Complexity and Liking as a Function of Expertise. *Music Perception*, 22(4):583–611, 2005.
11. Adi Ruppin and Hezy Yeshurun. MIDI Music Genre Classification by Invariant Features. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 397–399, Victoria, BC, Canada, October 2006.
12. Bob Snyder. *Music and Memory: An Introduction*. MIT Press, 2000.
13. Rainer Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Department of Information and Computing Sciences, Universiteit Utrecht, Netherlands, 2007.
14. Alexandra L. Uitdenbogerd and Justin Zobel. Manipulation of music for melody matching. In *ACM Multimedia*, pages 235–240, 1998.