Discovering Themes by Exact Pattern Matching

Lloyd Smith and Richard Medina
New Mexico Highlands University
Las Vegas, NM 87701
+1 505 454-3071
{las,richspider}@cs.nmhu.edu

1. INTRODUCTION

Content-based music information retrieval provides ways for people to locate and retrieve music based on its musical characteristics, rather than on more familiar metadata such as composer and title. The potential utility of such systems is attested to by music librarians, who report that library patrons often hum or whistle a phrase of music and ask them to identify the corresponding musical work [5, 7].

Content-based MIR systems operate by taking a musical query (i.e., a string of notes) from the user and searching the music database for a pattern closely matching the query. The search may be carried out by exhaustively matching the database [5] or by matching an index of n-grams created by passing a sliding window of length n over the database [2].

While these exhaustive search methods are adequate for relatively small music databases they do not scale well to large collections such as thousands of symphonies and other major works.

Fortunately, for large classical works, such as sonatas and symphonies, it is possible to avoid exhaustive search by using an index of themes. A theme, in classical music, is a melody that the composer uses as a starting point for development. A piece of music may have several themes; each of them will repeat and may be slightly changed (a "variation") by the composer on repetition. Using such an index greatly condenses the search on a database of classical major works. Furthermore, themes are the musical phrases likely to be remembered by listeners, so a theme index helps focus the search on the parts of the database most likely to match a query.

One well known theme index is that produced by Harold Barlow and Sam Morgenstern [1]. This is a print book containing approximately 10,000 themes from the classical music genre. Each theme is identified by composer, title of work and section of appearance (movement for symphony, act for ballet, overture for opera, and so forth). In addition to its print edition, this theme index can also be searched online [3].

Unfortunately, it is a monumental task to manually compile such a theme index. Because themes are, by definition, recurring patterns in a piece of music, it should be possible to automate the discovery of musical themes in order to create a theme index over a given database.

Our goal is to perform this automation – to analyze a piece of music and automatically determine its themes. Some work has been done on finding themes in music. Mongeau and Sankoff, for example, suggested the use of dynamic programming for finding recurring sections in a piece of music [6]. Their method, however, was somewhat cumbersome, relying on a closeness threshold to determine the beginnings and ends of recurring musical patterns.

2. DISCOVERING THEMES IN MUSIC

Because a theme, by definition, is a melody that the composer uses as a starting point for variation, most researchers have assumed that a theme discovery system must use approximate string matching [6].

Our approach takes the view that a theme dictionary may be constructed using an exact match of the musical sequence against itself. Our hypothesis is that a significant part of a theme is likely to repeat at least once, and that smaller chunks of a theme are likely to repeat multiple times. The basic idea is similar to that followed by Liu, et al. [4], but, where they build theme candidates by joining small repeating patterns into larger ones, we start with the longest repeating patterns and look for continuations and substrings.

Theme discovery is essentially a search for self-similarity in a piece of music. For that reason, we begin by creating a self-similarity matrix. For a musical piece of n notes, this is an $n \times n$ matrix representing where each interval exactly matches another interval in the piece – a repeated interval is represented by a *1* in the matrix. We use intervals in order to make our analysis independent of key. This does not make the analysis independent of mode – our algorithm is not likely to find repetitions of a major-key theme in minor mode, for example, or vice versa. If, however, a variation repeats multiple times, our algorithm will discover those patterns.

From the self-similarity matrix, we build a lattice showing all repeating patterns longer than a predetermined length and their relationships. At this point we are analyzing all repeating patterns of four or more notes. This lattice is further processed and used to determine which patterns to keep as candidate themes.

3. CASE STUDY

As a simple test of our algorithm, we used it to identify repeating patterns in Bach's Fugue No. 7, from the Well Tempered Clavier. We used only the top (soprano) part of the fugue – this is to find out whether our algorithm can find the theme without seeing its reiterations when the second and third voices enter. When processing the entire piece, we simply concatenate all parts to form one long sequence – this enables the algorithm to capture themes from repetitions in different voices, but does not attempt to discover themes split among more than one voice – the

algorithm is not expected to find a theme, for example, that starts in the first violin and migrates into the cello.

Figure 1 shows part of the lattice built from the top part of Fugue No. 7. As stated above, we ignore repeating patterns of fewer than 4 intervals.

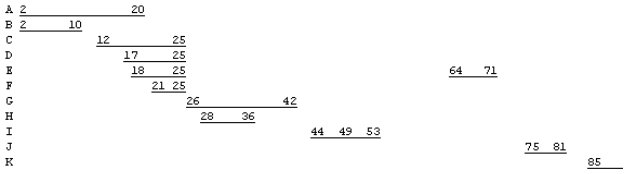


Figure 1. Partial lattice for fugue no. 7.

The lattice clearly shows a pattern of 18 notes, labeled A, near the beginning of the piece, starting with the second interval. This is not surprising, given the structure of a fugue, with the introductory statement of the subject. Pattern B is a substring of A, while pattern C overlaps A. Patterns D, E and F are substrings of C. This is only part of the lattice; pattern A repeats at position 311, B repeats at 93 and 311, C repeats at 103, and so forth. The lattice shows one repeat of pattern E at 64; E also repeats at 109 and 133. Pattern I overlaps itself, starting at position 44 and ending – and starting again – at position 49. A total of 20 repeating patterns were found.

We envision the theme index being searched by approximate search on user queries. For that reason, it is unecessary to keep patterns that are substrings of other patterns. Furthermore, we extend candidate themes by combining overlapping patterns. Figure 2 shows the lattice after discarding substrings and short patterns that occur only twice. Overlapping patterns A and C have been combined into one longer pattern. In fact, AC is the theme of the fugue – minus the first interval – as listed by Barlow and Morgenstern [1].

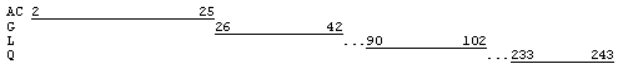


Figure 2. Lattice for fugue no. 7 after pruning.

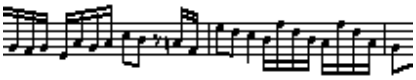
Pattern G appears twice in the fugue, it is long, and it has a substring that repeats, so it is included in the final list of patterns to be added to the theme dictionary.

Pattern L is mostly a substring of AC, and could be eliminated. However, it adds two (tied) notes to the beginning of AC and, at this point, is left in the list because of that extension. It is, in fact, a theme variation.

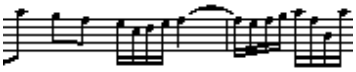
Pattern Q remains in the list because of its length. It appears only twice and has no repeating substrings. Inspection of Q shows that it is another variation on part of the theme, and it leads into a repetition of pattern C. Q is very similar to pattern B (the intervals up to the rest in AC), but introduces an E-natural in place of the expected E-flat.

Given the fact that a theme index is expected to be searched using approximate matching, it is likely that patterns L and Q should not be included – the first part of pattern AC is close enough to both of them to allow user queries to retrieve this particular fugue.

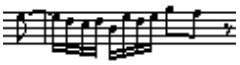
Figure 3 shows musical notation, extracted from the score, for patterns AC, G, L and Q,.



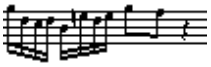
(a) Pattern AC



(b) Pattern G



(c) Pattern L



(d) Pattern Q

Figure 3. Candidate themes from Fugue No. 7.

4. CONCLUSION

This paper describes a method for automatically discovering themes in music. A program based on this algorithm can generate a theme index from a music database.

At this point, we have tested the algorithm using simple musical structures – namely, fugues. This provides a suitable beginning test because we can easily analyze whether the program is acting appropriately. In developing the algorithm, we have not made use of any musical knowledge regarding the structure of fugues, but have let the algorithm discover what it regards as themes. Our immediate goal now is to test the algorithm over a much wider range of music. A longer term goal is to produce a music analysis system based on the algorithm and to incorporate more sophisticated approximate matching to complement the base algorithm.

5. REFERENCES

[1] Barlow, H. and S. Morgenstern. *A Dictionary of Musical Themes*, Crown Publishers, NY, 1948.

[2] Downie, J. S. and M. Nelson. “Evaluation of a simple and effective music information retrieval method,” Proc ACM SIGIR 2000, 73-80, 2000.

[3] Huron, D. *Themefinder*, http://www.themefinder.org/.

[4] Liu, C. C., J. L. Hsu and A. L. P. Chen. “Efficient theme and non-trivial repeating pattern discovering in music databases,” Proc. IEEE Intl. Conf. on Data Engineering, 14-21, 1999.

[5] McNab, R. J., L. A. Smith, I. H. Witten and C. L. Henderson, “Tune retrieval in the multimedia library,” Multimedia Tools and Applications 10, 113-132, 2000.

[6] Mongeau, M. and D. Sankoff. “Comparison of musical sequences,” Computers and the Humanities 24, 161-175, 1990.

[7] Salosaari, P. and K. Jarvelin. “MUSIR -- a retrieval model for music,” Technical Report RN-1998-1, University of Tampere, Department of Information Studies, 1998.