NAME THAT TUNE!
AN INTRODUCTION TO MUSICAL INFORMATION RETRIEVAL

Jean Tague-Sutcliffe, Stephen Downie
School of Library and Information Science

Shane Dunne
Department of Computer Science

University of Western Ontario

In the beginning, computers, as their name implies, had numerical processing capability. The development of word processing and database systems gave them verbal processing capability and of graphics cards and software spatial processing capability. Now, with the widespread availability of sound cards and MIDI files, we may claim, as well, that computers have musical processing capability.

So, is it time to roll over Beethoven? Well, perhaps not yet. Though recent developments have given composers and performers some excellent musical tools, the power of the computer to access this new kind of language is still in its infancy. This paper looks at the potential of the computer for musical information retrieval, for getting information, not from text or database records, but from songs, symphonies, sonatas, sound tracks, and all the other musical forms. It also previews some of the work the authors are planning to carry out in this new IR area.

How to Give Computers Musical Intelligence

Giving a 386 or 486-based microcomputer musical capability is not particularly complicated or difficult. It involves only two or three steps:

1. Install a sound card, e.g. Sound Blaster Pro. A sound card contains a synthesizer capable of producing 4 or more voices of music or instruments simultaneously. For example, Sound Blaster Pro produces 22 voices of FM music. In addition, most sound cards contain a MIDI interface, with IN and OUT ports, to allow for transmission of musical information.

2. Attach speakers or a headphone set to the sound card.

3. If desired, attach a keyboard and a microphone for sound input, a MIDI sound module for improved output, and other MIDI instruments such as electronic pianos.

Because large amounts of MIDI-encoded and digitized music are available in CD-ROM format, it is highly recommended that a CD-ROM

drive form part of the configuration. An example of a musical microcomputing system is shown in Figure 1.

MIDI stands for Musical Instrument Digital Interface. It provides the means to encode musical performances for storage on digital media. For example, when a MIDI-equipped instrument such as a keyboard is played, it transmits data through its MIDI OUT port representing which keys are pressed, how hard they are pressed, which keys are released, etc. Connecting the instrument's MIDI OUT to the MIDI IN of the sound card makes it possible for the computer to record this information, using a type of program called a sequencer. When the sequencer subsequently reconstructs the recorded MIDI data stream via its MIDI OUT, and this is connected to the MIDI IN of the instrument, the original performance is reproduced.

This kind of sequencing/playback scheme admits many variations. For example, the MIDI data can be routed to a different instrument, yielding a playback of the same notes with different tonal characteristics. Most sound cards provide an internal synthesizer, obviating the need for an external MIDI device. Sequencers typically permit several "tracks" to be recorded separately and later played back simultaneously, by routing the MIDI data for different tracks either to different instruments or to a single "multi-timbral" instrument which can produce different-sounding notes simultaneously. The resulting system is analogous to a multi-track tape recorder.

The basic MIDI data format represents time only implicitly. A MIDI instrument is like a computer terminal; it sends codes through its MIDI OUT each time keys are pressed or released, and it plays notes immediately upon receipt of the corresponding codes via MIDI IN. The MIDI data stream is thus not a true sequence (in the mathematical sense) but rather a time series. Sequencer programs use a modified data representation which is a true sequence of event codes, each of which is tagged with a time value. There are now standard file formats for storing such event sequences on digital media, in the form of data files which have come to be called "MIDI files."

The MIDI file format is thus a standard format for sharing musical data files among different programs. Commercial music software packages offer advanced facilities such as interactive editing, extraction of parts from scores, and the ability to typeset highly readable music notation for use by human performers. An example of a MIDI edit screen in the software system Sequencer Plus is shown in Figure 2. MIDI files are supported by Windows with Multimedia Extensions (Versions 1 and 3.1) and also some DOS musical and multimedia packages. MIDI is not the only format for storing musical data: however, it is the most economical in terms of disk space. Hendall (1992), who provides an excellent introduction to MIDI, says that 200k of MIDI files will store about half an hour of

music.

Until now, MIDI files have been used principally for performance and composing music. Now, however, as more and more music becomes available in MIDI files, they present, to information scientists, opportunities for improving musical information retrieval, the provision of information concerning musical works.

Information Needs in the Field of Music

What kinds of questions have traditionally been asked about music? Do MIDI files have a potential for improving our ability to answer these questions? Can they provide resources which musicians and musical scholars have not, in the past, requested because their procurement seemed impossible?

Some queries in the field of music are text-based and parallel those in other fields:

1. List all compositions or all compositions of a certain form by a specified composer.

2. List all recordings of a specified composition or composer.

3. List all recordings of a specified performer.

4. Identify a song title given the first line of lyrics, or vice versa.

A good review of the role the computer has played in improving retrieval from textual catalogs of musical scores and discographies will be found in Duggan (1992). She points out, for example, that OCLC now contains catalog records for 606,000 scores and 719,000 sound recordings, and the Music Library CD-ROM published by Silver Platter contains more than 408,000 records for sound recordings. However, the ability to store the music itself, in MIDI files, provides us with the capability of answering queries beyond those served by a MARC-format-based catalog:

5. Given a composer, identify by the first few bars each of his or her compositions, or compositions of a certain type.

6. Given a melody, for example the tune of a song or the theme of a symphony, identify the composition from which it comes.

The first of these two types of queries has traditionally been answered by means of a printed incipit index, a listing of the beginning bars of the scores of the works in a particular collection. An example of a printed incipit index is shown in Figure 3.

The second type of query has traditionally been answered by thematic indexes to musical compositions. An example of such an index (Barlow and Morgenstern, 1949) is shown in Figure 4. The book contains a few bars of one or more themes from 10,000 musical compositions, arranged by composer. A 'Notation Index', in the back of the book, permits an inquirer to look up a sequence of six to eight notes, transposed to the key of C, in an alphabetical listing of transposed 'themes' to identify the composition in which it occurs.

A variation on satisfying queries about musical themes is the 'Humline', described in a recent issue of the Library Association Record (v.94, 1992, 155). The Birmingham (U.K.) Library Service provides a telephone-based information service for people who want to identify pieces of music. Inquirers hum tunes into an answering machine and a librarian undertakes a manual search to locate it. No actual index is involved.

Incipit and thematic indexes can be automated by developing appropriate data structures for accessing musical compositions in MIDI form by their first few bars or themes, just as books and journal articles are accessed by keywords. Users can then input the first section or theme they wish to identify and a search algorithm locates it using this data structure. However, musical information retrieval has the potential for answering more extensive queries, involving the 'full-text' of musical compositions. Such queries include:

- In which compositions can we find the following note sequence anywhere in the composition?
- Which composers have used the following harmonic progression?
- Which composers have used the following combination of instruments in the orchestration of a passage?

The Design of Music Databases

We will define a true music database as one which contains, in addition to the bibliographic information one might find in an online catalog for musical scores or recordings, pointers to MIDI (or similar format) files recording performances of each composition, and the MIDI files themselves. Queries to this database can be described, most generally, as consisting of patterns of either text strings or musical fragments, or both. The musical fragments represent sequences of notes, some of which may overlap in time (polyphony). Information about note duration and loudness may also be a part of the fragment.

Retrieval from the textual attributes of the database (composer, title, identification number, key, performers, dates, etc.) would

be similar to that from existing bibliographic databases and could involve the usual well-known methods of boolean or weighted keyword searching of these attributes. Retrieval from the MIDI files, however, presents new challenges.

There are four approaches to the design of a music database where queries will be in the form of musical fragments: linear scanning of the MIDI files, construction of an inverted index to all musical fragments in the MIDI files, construction of a hash table for musical fragments in the MIDI files, and construction of a bit vector to represent the musical fragments in each MIDI file. These approaches differ in the complexity of the retrieval algorithms.

Linear scanning means that each MIDI file is searched, in turn, from beginning to end, for the query fragment. To scan a sequence of MIDI files whose total length could be represented by the number n (e.g., n pages, n songs of similar length, n bars of music, n 'records') for a musical fragment would require a search time proportional to n, or a search complexity of O(n).

Inverted indexes are frequently used as the access mechanism for keyword searches of bibliographic databases. The index contains, in sorted order, all the unique strings of up to a specified length which appear in the database, together with a pointer to a postings list, the list of locations within the bibliographic files, where the sought for string may be found. The index is accessed by a binary "divide and conquer' search algorithm. Indexing musical databases means we must, first of all, determine what length of fragment will be indexed and whether any fragments will be ignored in the index (like stopwords in a textual database index).

With musical fragments, however, there are other concerns. A decision must be made as to whether information about note duration and other aspects represented in the MIDI file such as loudness or pitch bending will be included. The user may be interested in a theme or tune regardless of the key. Thus, for identification purposes, it is more useful to index sequences of musical intervals, rather than musical notes. Thus, the beginning of the nursery rhyme "Twinkle twinkle little star' might be represented as the note sequence CCGGAAGFFEEDDC or the sequence GGDDEEDCCBBAAG; these could both be represented by the interval sequence 07020 (-2)0(-2)0(-1)0(-2)0(-2), where the entry in the sequence indicates the number of semitones between one note and the next.

Polyphonic music presents special challenges. For example, if the music contains chords, rather than simply melodies, there are many more possible matches for a query fragment, since the match may begin in one 'voice' and then move to a lower or higher voice.

As in textual searching, inverted files would save time in searching musical databases. If the size of the musical database can be represented by the number n, as above, the search complexity

is O(log n). However, the improved search speed is obtained at a cost; if there are a total of m distinct fragments in the index, it will take a total time of order O(mlogm) to build. If MIDI files are constantly being added to the database, it will be necessary, periodically, to rebuild the index; thus, the building time is a major concern.

Hashing is an approach in which the search time to find a fragment in the MIDI files is independent of the total size of the files, i.e., the complexity is O(1). A possible hash function is one which maps each of the set of possible musical interval sequences of a fixed length k into a finite subset of integers {0,1, ... q}. For example, suppose there are c possible musical intervals and that $s=i_1 i_2 \ldots i_k$ represents the length-k interval sequence. The function

$$H(s) = \Sigma_j \; i_j c^{k-j},$$

for j=1 to k, will always be an integer between 0 and $q=c^k-1$. The hash table is an array, held in memory. The hth element in the table is a 'postings list' or a pointer to a postings list of the files which contain musical fragments whose hash function value H(s)=h.

If the query fragment contains more than k intervals, it is divided into sequences of length-k fragments. Postings lists are obtained for each length-k fragment and the final set of locations of the fragment is obtained by taking the intersection of these lists.

A problem with hashing is false drops. Not all of the musical selections in the postings list will actually contain the query fragment. Thus, a scan of these selections will be necessary to determine which ones are actual hits. However, if the hashing process is a reasonably good one, the set of MIDI files in the hit list will be much smaller than the total set of MIDI files in the database.

The MIDI format provides for 128 different pitch values, one semitone apart. There are therefore 255 possible intervals, from -127 to 127. For the purposes of indexing and hashing, it may be desirable to partition this set into a smaller number of classes. For example, we could reduce the number of interval classes to 23 (-11 to +11) by taking the original interval and computing the signed remainder after dividing by 12. Musically, this identifies notes having the same name (e.g. C-sharp) in different octaves.

The advantage of interval classification is that it speeds up search of the index; the disadvantage is that it creates more false drops to eliminate in the final processing. Also, as with truncation in text searching, it may improve recall.

The final database access design to be presented here is the bit vector approach. Instead of an index or hash table, a bit vector or array of binary values (a signature) is constructed for each MIDI file. This vector indicates, for every possible length-k interval class sequence, whether or not that sequence occurs in the file. To process a query, one would construct a bit vector for the query fragment itself and intersect this with each of the per-file bit vectors. Thus, search time would be proportional to the size of the number n of MIDI files in the database and complexity would be O(n). Bit vector operations are extremely fast and the bit vectors themselves are space-efficient. However, for large collections of MIDI files, the space required for the bit vectors might still be considerable. The space requirement for bit vectors is proportional to the number C^k-1 of length-k interval classes; the number of bits required per file will be exactly the same as the number of entries in a hash table constructed according to the scheme described earlier.

Research Agenda

There are a number of questions yet to be answered about musical information retrieval, given the music database framework described in the last section. The authors are in the initial stages of a research project, funded in part by NSERC, to examine some of these questions. Specifically, we will be investigating the following:

- User Needs Assessment: What kinds of user needs among composers, musical performers, musicologists, and lay people, can be answered from musical databases?

- Access Method Assessment: Under what conditions of memory and database size and musical fragment distribution are each of the four access methods described above optimal? What is the effect of the values of k (the length of the interval or interval-class sequences which may be directly searched) and c (the number of classes into which intervals are classified) on index size, search time, and numbers of false drops?

- What length of musical fragment and what parameters--pitch or interval sequence, note duration, or other MIDI recorded features--must be included in a query description in order to identify the relevant musical fragments desired by different classes of users of a music database? How can nonrelevant retrievals be reduced?

- What kind of output, what parts of the retrieved musical composition and in what form, what other attributes, do users of a music database need?

We hope, at a future meeting of the Canadian Association for Information Science, to bring you some answers to these questions.

References

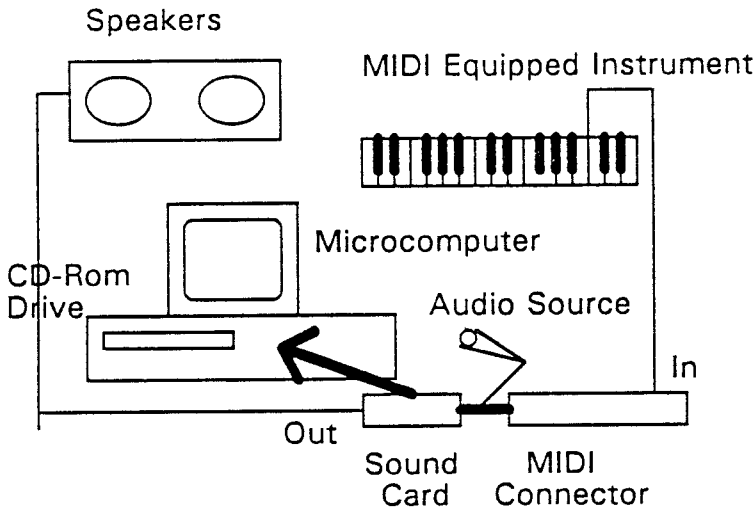
H. Barlow and S. Morgenstern, A Dictionary of Musical Themes, Benn, 1949.

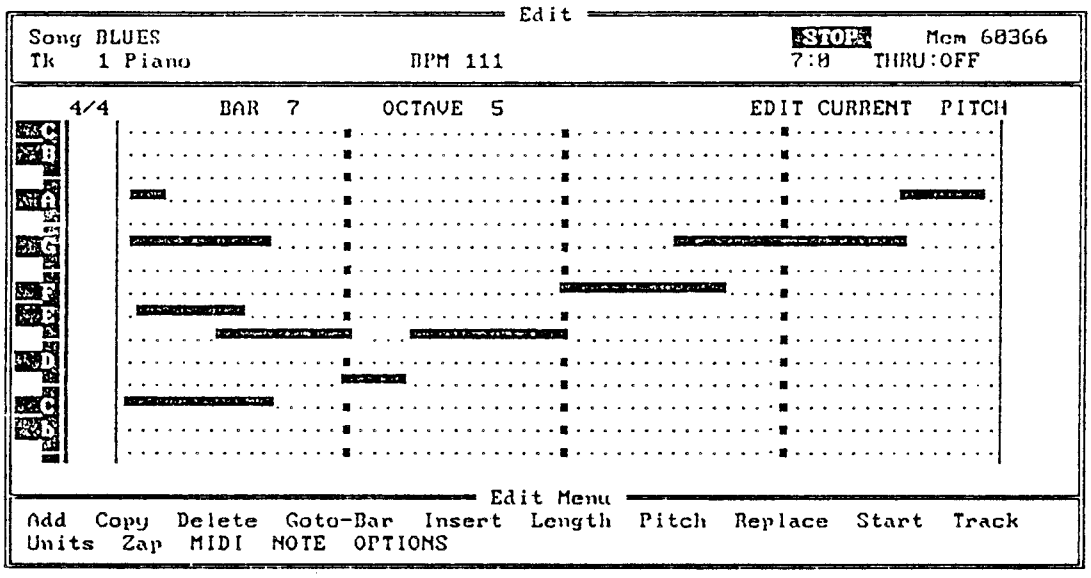
M.K. Duggan, "Electronic information and applications in musicology and music theory," Library Trends 40(4), 1992, 756-80.

J.S. Edson, Organ--Preludes; an Index to Compositions on Hymn Tunes, Chorales, Plainsong Melodies, Gregorian Tunes and Carols, Scarecrow Press, 1970.

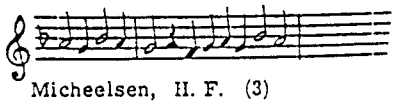
R. Kendall, "MIDI goes mainstream," PC Magazine 11(6), 1992, 183-218.

"There's a kind of a hum--all over the world", Library Association Record 94, 1992, 155.


Speakers
MIDI Equipped Instrument
Microcomputer
CD-Rom
Drive
Audio Source
In
Out
Sound
Card
MIDI
Connector


Edit
Song BLUES
Tk 1 Piano
BPM 111
STOP
Mem 68366
7:0 THRU:OFF
4/4
BAR 7
OCTAVE 5
EDIT CURRENT PITCH
Edit Menu
Add Copy Delete Goto-Bar Insert Length Pitch Replace Start Track
Units Zap MIDI NOTE OPTIONS

Brich uns, Herr, das Brot (Micheelsen)



Metzger, H. A.

Micheelsen, H. F. (3)

Brightly Beams Our Father's Mercy (Bliss)

Thompson, V. D.

Variant:
Lower Lights

Wilson, R. C.

Bring a Torch (French)

Rogers, S. E.

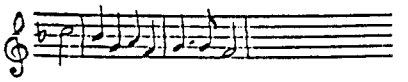
Williams, D. H.

Variant:
Un Flambeau

Bingham, S.

Hulse, C. Van

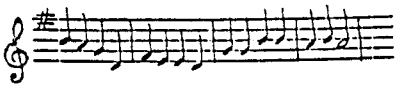
Bristol (Ravenscroft)



*Cameron, G.
Dyson, G.
Groves, R.
Hunt, W.
Lang, C. S.

Palmer, C. C.
West, J. E.
Westrup, J. A.
Willan, H.

Brocklesbury (Barnard)



Powell, R. J.

Broedre og soestre, vi skilles (Simonson)

Frandsen, H. B.

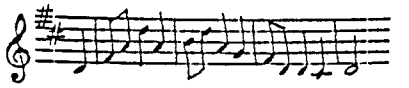
Wuertz, J.

Bromsgrove (Dyer)

Rowley, A.

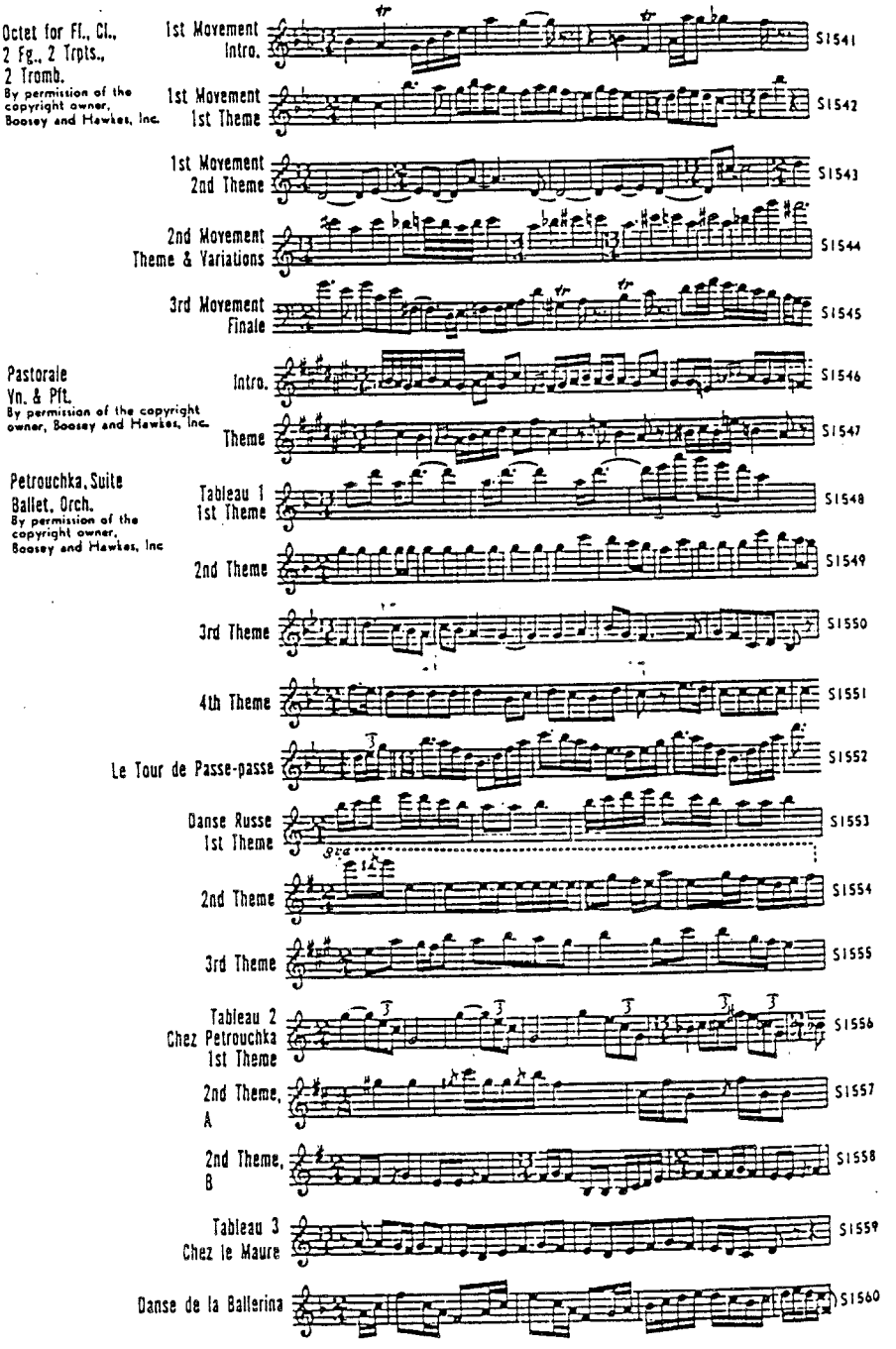
Brorson
See: Kender du den (Berggreen)

Brother James Air (Bain)



Darke, H. E.
Owens, S. B.
Wright, M. S.

Variant:
Marosa


Octet for Fl., Cl., 2 Fg., 2 Trpts., 2 Tromb.
By permission of the copyright owner, Boosey and Hawkes, Inc.
1st Movement Intro.
S1541
1st Movement 1st Theme
S1542
1st Movement 2nd Theme
S1543
2nd Movement Theme & Variations
S1544
3rd Movement Finale
S1545
Pastorale Vn. & Pft.
By permission of the copyright owner, Boosey and Hawkes, Inc.
Intro.
S1546
Theme
S1547
Petrouchka, Suite Ballet, Orch.
By permission of the copyright owner, Boosey and Hawkes, Inc
Tableau 1 1st Theme
S1548
2nd Theme
S1549
3rd Theme
S1550
4th Theme
S1551
Le Tour de Passe-passe
S1552
Danse Russe 1st Theme
S1553
2nd Theme
S1554
3rd Theme
S1555
Tableau 2 Chez Petrouchka 1st Theme
S1556
2nd Theme, A
S1557
2nd Theme, B
S1558
Tableau 3 Chez le Maure
S1559
Danse de la Ballerina
S1560

NOTATION INDEX