Pattern Discovery and Pattern Matching in Polyphonic Music and Other Multidimensional Datasets

David Meredith
Department of Computing, City University, London.
dave@titanmusic.com

Geraint A. Wiggins
Department of Computing, City University, London.
geraint@soi.city.ac.uk

Kjell Lemström
Department of Computer Science, University of Helsinki.
klemstro@cs.helsinki.fi

Music, Minds, Machines (M^3) Seminar,
Faculty of Music, University of Edinburgh.

Tuesday, 12 June 2001.

1. Pattern Discovery and Pattern Matching in Polyphonic Music and Other Multidimensional Datasets

1. The importance of repetition in music.
2. Not all repetition is significant.
3. The variety of musically significant repeated patterns.
4. Other approaches to pattern discovery in music.
5. `SIA`: Finding maximal repeated patterns.
6. `SIATEC`: Finding all the occurrences of each maximal repeated pattern.
7. `MU`: Selecting the musically interesting repeated patterns.
8. Some preliminary results.
9. Possible uses for a pattern discovery algorithm for polyphonic music.
10. `SIA(M)ESE`: Flexible pattern matching in multidimensional datasets.

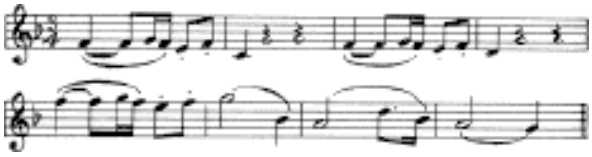
1. Pattern Discovery and Pattern Matching in Polyphonic Music and Other Multidimensional Datasets

1. We're going to be talking to you about pattern discovery and pattern matching in polyphonic music and in multidimensional datasets in general.
2. I'll talk for about half an hour on the work that I've done on pattern discovery, focusing on `SIA` and `SIATEC`, which are two new efficient pattern-discovery algorithms that I've developed over the past year or so in collaboration with Geraint Wiggins and Kjell Lemström.
3. Geraint will then explain how `SIA` can be adapted and generalised to produce an efficient and flexible pattern-*matching* algorithm which he calls `SIA(M)ESE`.
4. We're both going to concentrate on how these algorithms can be used for processing *music* data but you should be aware that the algorithms are, in fact, quite general and could be applied to any data that can appropriately be represented in the form of a multidimensional dataset (that is, a set of points in a k-dimensional space.)
5. `SIA` and `SIATEC` discover maximal repeated structures in multidimensional datasets. I'm going to begin by arguing that the discovery of significant repetition in a piece of music is an essential step in the process by which a listener achieves a rich and satisfying interpretation of the piece.
6. I'll then go on to show that, although musically significant repetitions are extremely important, not all the structural repetitions that occur in a piece are interesting and significant.
7. I'll then attempt to show that characterising the class of interesting repeated patterns is no simple matter because this class of patterns is so diverse.
8. I'll briefly review other approaches to pattern discovery in music and then I'll show how the shortcomings of these other approaches can be overcome by abandoning the notion of representing music as one-dimensional strings in favour of a multidimensional representation.
9. I'll then describe the principle underlying my `SIA` algorithm that allows it to discover the maximal repeated patterns in a k-dimensional dataset of size n in a worst-case running time of $O(kn^2 \log_2 n)$.
10. And then I'll describe the basic idea behind my `SIATEC` algorithm which finds all the occurrences of all the maximal repeated patterns in a worst-case running time of $O(kn^3)$.
11. Our experiments show that the set of patterns generated by `SIA` for a dataset representing a musical score typically contains the musically significant repeated patterns. However, the output usually also contains many patterns that are *not* musically interesting. So at the moment I'm developing a program called `MU` which evaluates the

output of SIATEC and attempts to isolate the musically most interesting repeated patterns.

12. MU and SIATEC together form a system that I call MUSIATEC and I'll show you what MUSIATEC generates for a couple of different datasets.
13. I'll finish up by just mentioning some of the potential applications of SIA and SIATEC that we've proposed in our patent application.
14. I'll then hand over to Geraint who will describe his pattern-matching algorithm SIA(M)ESE.

2. The importance of repetition (1)



[Grouping is] an auditory analog of the partitioning of the visual field into objects, parts of objects, and parts of parts of objects.

(Lerdahl and Jackendoff, 1983, p.36)

Grouping structure theory = 5 GWFRs + 7 GPRS

GPR 6 (Parallelism) Where two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups.

(Lerdahl and Jackendoff, 1983, p.51)

[Parallelism is] also the major factor in all large-scale grouping. For example, it recognizes the parallelism between the exposition and the recapitulation of a sonata movement, and assigns them parallel groupings at a very large level, establishing major structural boundaries in the movement.

(Lerdahl and Jackendoff, 1983, p.52)

[T]he importance of parallelism in musical structure cannot be overestimated. The more parallelism one can detect, the more internally coherent an analysis becomes, and the less independent information must be processed and retained in hearing or remembering a piece.

(Lerdahl and Jackendoff, 1983, p.52)

2. The importance of repetition (1)

1. I'd like to begin by reminding you of the importance of repetition in musical structure.

2. One of the four main components of Lerdahl and Jackendoff's *Generative Theory of Tonal Music* is devoted to *grouping structure* (Lerdahl and Jackendoff, 1983, 36–67).

3. They describe the task of "grouping" a musical surface as being [READ FROM SLIDE] "an auditory analog of the partitioning of the visual field into objects, parts of objects, and parts of parts of objects" (Lerdahl and Jackendoff, 1983, 36).

4. Lerdahl and Jackendoff's grouping structure theory consists of five "grouping well-formedness rules" (GWFRs) and seven "grouping preference rules" (GPRs). The GWFRs "define the formal notion *group* by stating the conditions that all possible grouping structures must satisfy" (Lerdahl and Jackendoff, 1983, 37). The grouping preference rules, on the other hand, "establish which of the formally possible structures that can be assigned to a piece correspond to the listener's actual intuitions."

5. Of the seven GPRs, numbers 1 to 4 deal with local detail and numbers 5 to 7 deal with larger-scale grouping. Of the three larger-scale grouping rules, GPR 6, the 'Parallelism' rule (Lerdahl and Jackendoff, 1983, 51) stands out as being of special relevance to the subject of repetition in music. [READ FROM SLIDE]

6. Lerdahl and Jackendoff point out that "GPR 6 says specifically that parallel passages should be analyzed as forming parallel parts of groups rather than entire groups" and explain that it is stated this way so that it can "deal with the common situation in which groups begin in parallel fashion and diverge somewhere in the middle, often in order for the second group to make a cadential formula".

7. This example here (reproduced from Lerdahl and Jackendoff, 1983, 51) demonstrates this. Bar 3 is an exact repetition of bar 1 and bar 5 is an exact transposition up an octave of bar 1. Bars 1, 3 and 5 are therefore segments that "can be construed as parallel" and should therefore, according to GPR 6, "preferably form parallel parts of groups." Since bar 1 can only form the beginning of a group (it begins the piece), bars 3 and 5 must therefore each form the beginning of its group.

8. This example demonstrates that the recognition of parallelism—and, in particular, the identification of certain instances of exact and exact transposed repetition—is "important in establishing intermediate-level groupings."

9. Lerdahl and Jackendoff (1983, 52) point out that GPR 6 (and, by implication, the identification and cataloguing of repetition) is [READ FROM SLIDE]

> also the major factor in all large-scale grouping. For example, it recognizes the parallelism between the exposition and the recapitulation of a sonata movement, and assigns them parallel groupings at a very large level, establishing major structural boundaries in the movement.

10. So, in Lerdahl and Jackendoff's (1983, 52) opinion, [READ FROM SLIDE]

> the importance of parallelism in musical structure cannot be overestimated. The more parallelism one can detect, the more internally coherent an analysis becomes, and the less independent information must be processed and retained in hearing or remembering a piece.

3. The importance of repetition (2)

Analysis is the means of answering directly the question 'How does it work?'. Its central activity is comparison. By comparison it determines the structural elements and discovers the functions of those elements. Comparison is common to all kinds of musical analysis—feature analysis, formal analysis, functional analysis, information-theory analysis, Schenkerian analysis, semiotic analysis, style analysis and so on: comparison of unit with unit, whether within a single work, or between two works, or between the work and an abstract 'model' such as sonata form or a recognized style. The central analytical act is thus the test for identity.

(Bent and Drabkin, 1987, p.5)

Je... choisirai, comme principal critère de division, la *répétition*. Je partirai de la constatation empirique du rôle énorme joué en musique, à tous les niveaux, par la répétition...

[I... will choose to use *repetition* as my principal criterion for segmentation. I will take as my starting point the empirical observation that repetition plays an enormously important rôle in music at all levels...]

(Ruwet, 1972, p.111)

c'est sur la répétition—ou l'absence de répétition—qu'est fondé notre découpage. Lorsqu'une suite de sons est énoncée à deux ou plusieurs reprises, avec ou sans variante, elle est considérée comme une unité. Corollairement, une suite de sons énoncée une seule fois, quels que soient sa longueur et le nombre apparent de ses articulations (notamment les silences) est considérée elle aussi comme une unité...

[Our segmentation is based on repetition—or the absence of repetition. When a sequence of sounds is stated two or more times, with or without variation, it is considered to be a unit. It follows from this that a sequence of sounds stated once, regardless of how long it may be or how many points of articulation (particularly rests) it may seem to contain is also considered to be a unit...]

(Rouget, 1961, 41)

3. The importance of repetition (2)

1. Lerdahl and Jackendoff are by no means the only authors who have emphasized the importance of identifying repetitions (or 'parallelism' as they call it) in the process of achieving a rich interpretation of a musical work.

2. Music analysts have been stressing the importance of identifying repeated structures for at least 40 years. In his masterly *New Grove* article on the subject, Ian Bent says [READ FROM SLIDE].

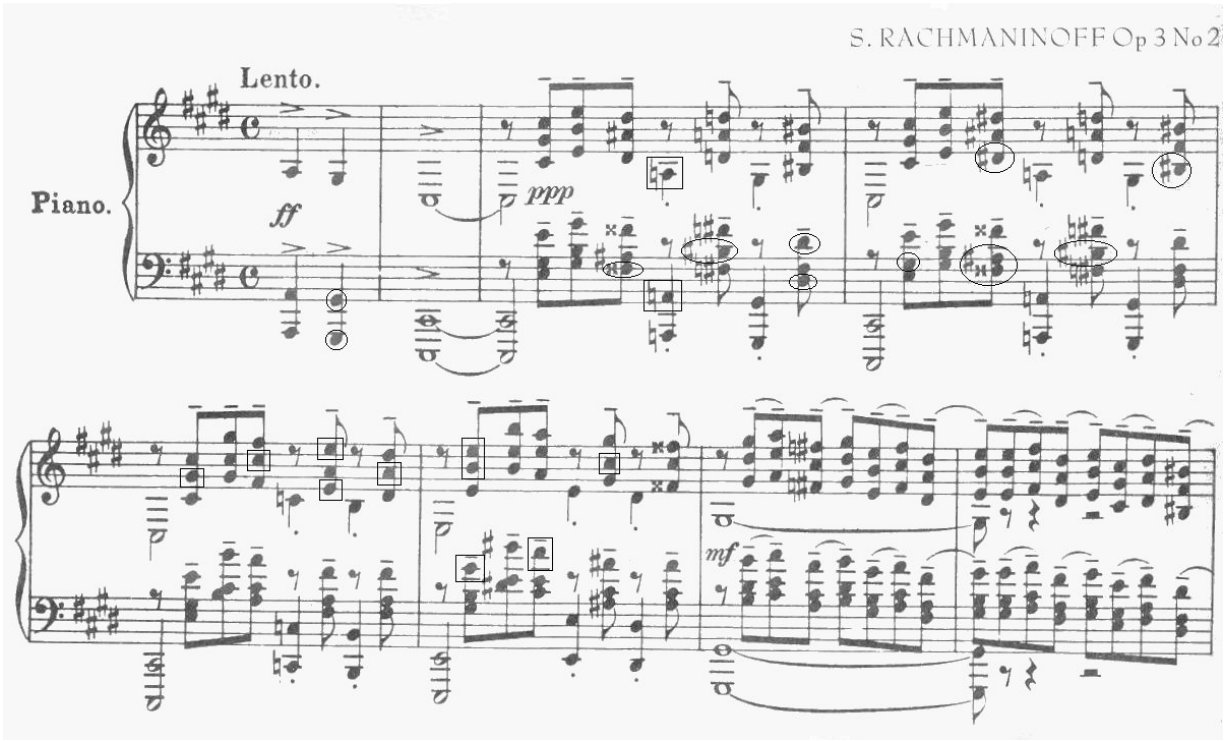
3. Although, as Bent says, the act of comparison and testing for identity is common to most approaches to music analysis, identifying repeated structures is absolutely fundamental to *musical semiotics*, a field of analysis pioneered by Nicolas Ruwet (1972) and developed further by a number of analysts including, in particular, Jean-Jacques Nattiez (1975).

4. Ruwet first described the basic method of semiotic analysis in 1966 in his paper *Méthodes d'analyse en musicologie* (Ruwet, 1966). As Ruwet describes it, the method can only be applied to monophonic music. The aim of the method is to take a monophonic musical surface and derive from this surface a 'syntax' that can account for its structure. The first step in this process is to segment the surface and Ruwet uses the principle of repetition as his 'principal criterion for segmentation'. As he says, [READ FROM SLIDE].

5. In doing this,Ruwet was in fact following a precedent set even earlier in 1961 by Gilbert Rouget in his study of African chromaticism. Rouget says [READ FROM SLIDE].

6. So it seems fair to conclude that many music analysts and psychologists of music agree that the identification and cataloguing of significant instances of repetition within a musical work forms an important early step in the process of achieving a rich interpretation of that work.

4. Not all repetition is musically significant



The pattern consisting of the notes in square boxes is an exact transposed repetition of the pattern consisting of the notes in elliptical boxes.

4. Not all repetition is musically significant

1. I'd now just like to demonstrate that although structurally significant repetitions are fundamental to a listener's understanding of piece of music, not all the repetitions that occur in a piece are interesting and significant.

2. For example, here we have the first few bars of Rachmaninoff's famous *Prelude* in C sharp minor, Op.3, No.2. The pattern consisting of the notes in round boxes is repeated 7 crotchets later, transposed up a minor ninth to give the pattern consisting of the notes in square boxes. [SHOW ON SLIDE.]

3. This is what these few bars sound like [PLAY RACH-bs1-6.MID].

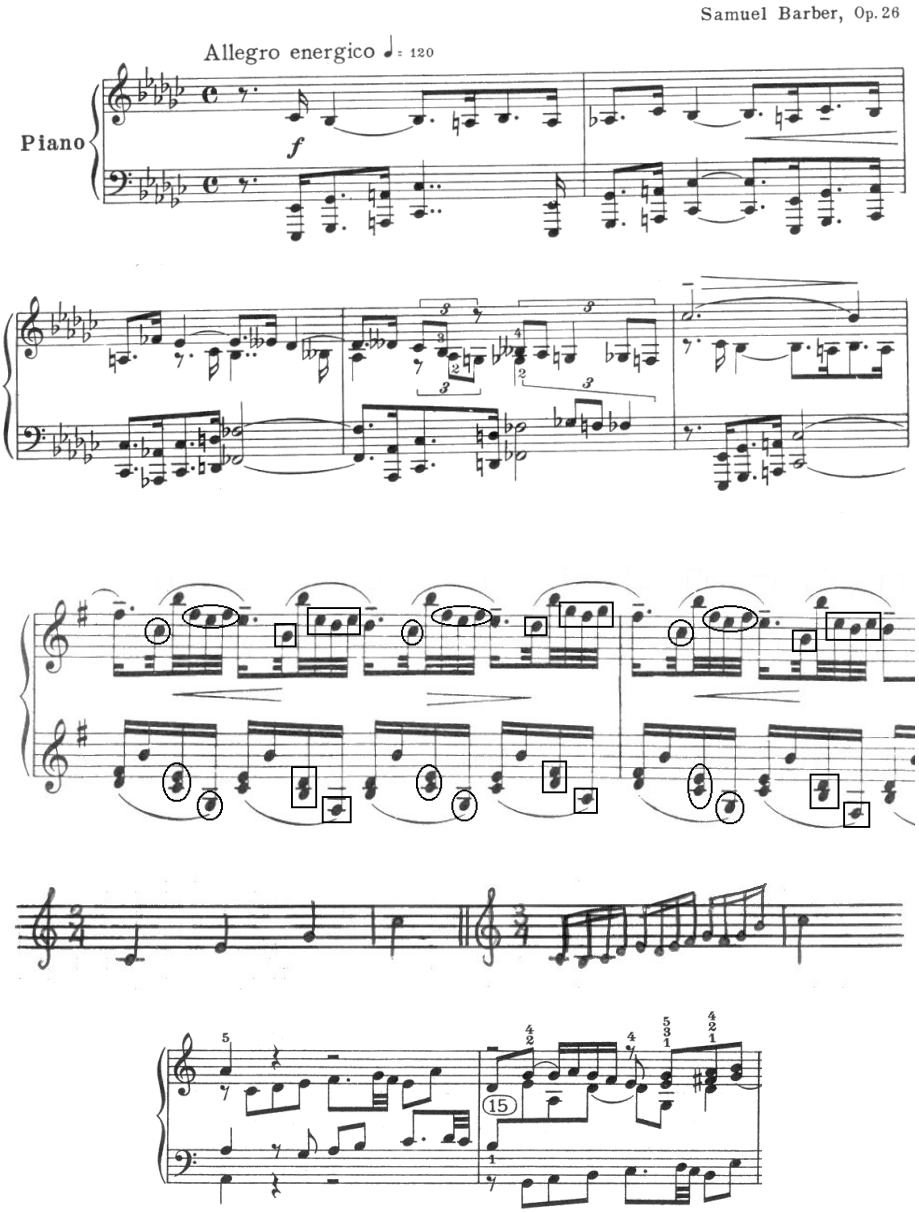
4. Now I'm going to play the same bars with the pattern notes emphasized. [PLAY BADPATTERN.MID].

5. Clearly, this repetition is just an artefact that results from the other musically significant repetitions that are occurring in this passage such as, for example, the exact repetition of bar 3 in bar 4.

6. In fact, it turns out that typically the vast majority of exact repetitions that occur within a piece of music are *not* musically interesting.

7. Our task, therefore, involves formally characterising what it is about the interesting structural repetitions that distinguishes them from the many exact repetitions that the expert listener and analyst do not recognize as being structurally significant.

5. Different types of musically interesting repeated patterns



5. Different types of musically interesting repeated patterns

1. It would probably not be going too far to say that for a musical pattern to function as a perceptually significant structural unit it must either be repeated in some way within the music or it must be constructed out of repeated units. However, characterising the class of 'perceptually significant' repeated patterns is no simple matter because this class of patterns is so diverse. In fact, it's probably best characterised as a set of sub-classes.

2. A musically significant repeated pattern may be just a very small motif, consisting of no more than a few notes. However, if this is the case, then it will typically be repeated frequently over the course of a piece. This occurs, for example, in much of Brahms' so-called 'monothematic' music. It also occurs here in the opening bars of Barber's *Sonata for Piano* where the left-hand motif is repeated over and over again. Here's what it sounds like [PLAY BARBER.MID]. And here's what it sounds like with that repeated motif emphasized. [PLAY BARBERMODIFIED.MID].

3. On the other hand, in a sonata form movement there may be large chunks of the exposition that are repeated in the recapitulation, each repeated chunk consisting of maybe hundreds of notes.

4. So a significant repeated pattern may consist of just a few notes or a significant proportion of all the notes in a piece. We *can* say, however, that for a small pattern to be significant it has to be repeated frequently whereas the largest interesting repeated patterns are usually only repeated once or twice.

5. In many simple songs, the significant repeated patterns are often purely monophonic. However, in polyphonic music and music like piano music where the voicing is not explicitly represented, a significant repeated pattern may consist of notes from several different voices that overlap each-other in time. An example of this is the repeat of bar 3 in the Rachmaninoff *Prelude* that I just showed you [SHOW SLIDE NO 4 AGAIN]

6. A significant repeated pattern may contain all the notes that occur within a particular time interval during the piece—as in bars 3 and 4 here—or it may contain only some of the notes that occur during the time interval that it spans—as in this Barber example [PUT SLIDE 5 BACK ON AGAIN].

7. If the pattern contains only some of the notes that occur during the time interval that it spans then these notes may include all the notes in one of the voices or all the notes in two or more of the voices (as it sort of does here in the Barber example). Alternatively, the pattern may contain some notes from one voice and some notes from another voice as happens in this example taken from another Rachmaninoff *Prelude*, this time Op.32 No.5. This is what the passage sounds like [PLAY RACH325.mid] and this is the same passage with the repeated pattern emphasized [PLAY RACH325emph.mid].

8. A pattern can also be repeated in an ornamented form. For example, here we have a rising C major arpeggio that's elaborated using a technique known as 'diminution' which involves inserting shorter ornamental notes in between the main notes of the pattern. This sounds like this [PLAY ARPEGGIO.MID]

9. Or occurrences of the pattern may overlap in time as occurs when you get a *stretto* in a contrapuntal piece as shown here in this extract from a fugue by Bach which sounds like this [PLAY STRETTO.MID].

6. Other pattern discovery algorithms for music

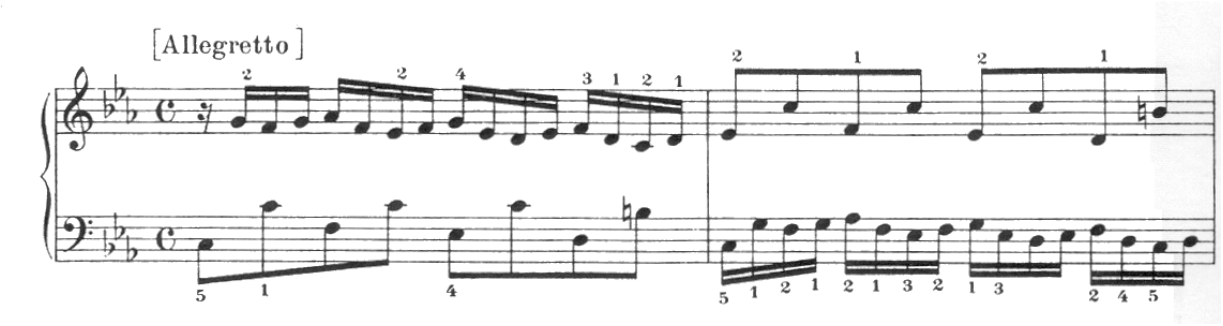
- Rolland (1999) (FlExPat)
  - Only deals with monophonic music.
  - Can only find patterns whose sizes lie within a user-specified range.
  - Uses edit-distance approach to approximate matching.
- Hsu et al. (1998)
  - Only works for monophonic music.
  - Cannot find transposed repetitions.
  - Slow (worst-case running time of $O(n^4)$).
  - Does not allow for gaps.
- Cambouropoulos (1998)
  - Does not allow for gaps.
  - Patterns must be bounded by pre-computed 'local boundaries'.
  - Only works for monophonic music.
  - Uses edit-distance approach to approximate matching.

6. Other pattern discovery algorithms for music

1. As you've probably gathered by now, my goal is to develop an algorithm that can discover all the interesting repeated patterns in any piece of music given to it as input.
2. But as I've just shown, the class of repeated patterns that might be of interest to an analyst or expert listener contains a wide variety of different types of pattern.
3. It seems that most previous attempts to develop a pattern discovery algorithm for music have been based on string-matching techniques.
4. An example of such an algorithm is Pierre-Yves Rolland's FlExPat program (Rolland, 1999). This program can find approximately repeated patterns within a monophonic source. However it suffers from three weaknesses.
   (a) First, it can only deal with monophonic music.
   (b) Second, it can only find patterns whose sizes lie within a user-specified range (and if the range is defined so that it allows patterns of any size, the overall worst-case running time goes up to at least $O(n^4)$).
   (c) Third, like most string-based approaches to approximate pattern matching, it uses the edit-distance approach. Unfortunately, such an approach is not typically capable of finding a repetition like the one shown here [ARPEGGIO] because the edit distance between these two occurrences is actually quite large owing to the high number of insertions required to transform the plain version into the ornamented one. A program like Rolland's regards two patterns as being similar if the edit distance between them is less than some threshold k. However, for these two patterns to be considered 'similar' by Rolland's algorithm, this value of k would have to be set to 9. Unfortunately, this value would in general be too high because the program would then start regarding highly dissimilar patterns as being similar.
5. Hsu et al. (1998) have also described a pattern discovery algorithm for music. Their algorithm is based on dynamic programming but it suffers from a number of serious weaknesses:
   (a) First, again, it only works for monophonic music (or polyphonic music in which each voice is represented as separate string which means that it can only find patterns that are wholly contained within one voice.)
   (b) Second, it is not capable, as described, of finding transposed repetitions.
   (c) Third, it has a worst-case running time of $O(n^4)$ which means it's too slow to be used for analysing large pieces.
   (d) Finally, it cannot find patterns 'with gaps'. That is, it can only find a pattern if it contains all the notes in the piece that occur during the time interval spanned by the pattern.

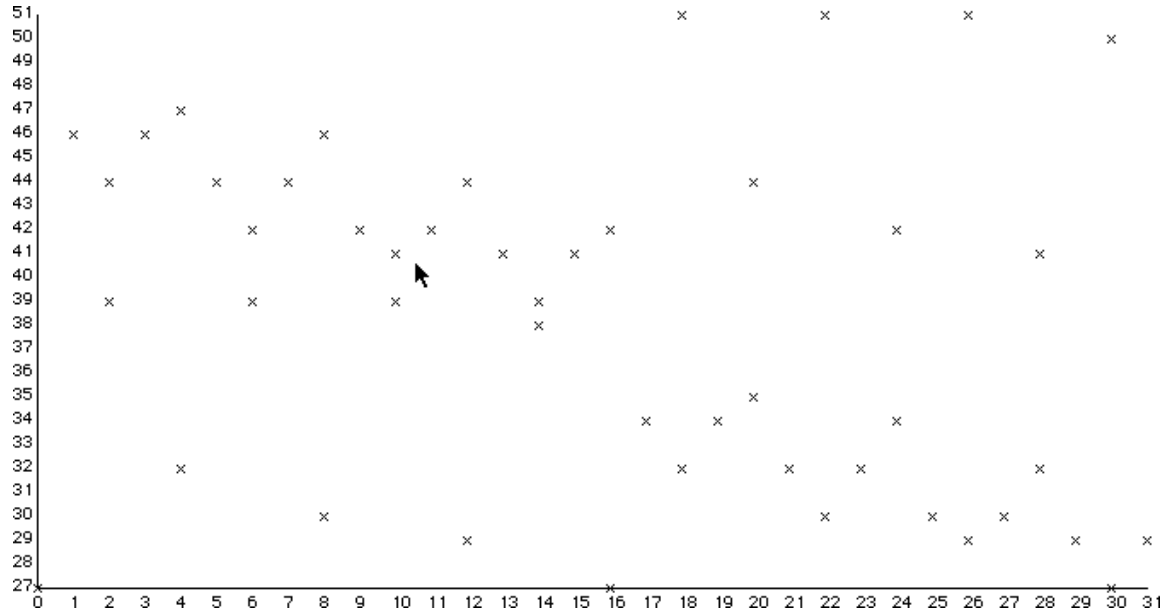
6. Cambouropoulos' (1998) *General Computational Theory of Musical Structure* also contains a pattern discovery component. This program performs approximate pattern discovery like Rolland's FlExPat and uses a novel algorithm for categorising the patterns found. However, again, this program only works for monophonic music and cannot find patterns with gaps. Moreover, the patterns found must be bounded by pre-computed 'local boundaries'. Cambouropoulos algrithm also uses the edit-distance approach to approximate matching which means that it would be incapable of finding repetitions like this arpeggio example.

7. Representing music using multidimensional datasets (1)



⟨onset time, chromatic pitch, morphetic pitch, duration, voice⟩

{ $\langle 0, 27, 16, 2, 2\rangle$, $\langle 1, 46, 27, 1, 1\rangle$, $\langle 2, 39, 23, 2, 2\rangle$, $\langle 2, 44, 26, 1, 1\rangle$, $\langle 3, 46, 27, 1, 1\rangle$,
$\langle 4, 32, 19, 2, 2\rangle$, $\langle 4, 47, 28, 1, 1\rangle$, $\langle 5, 44, 26, 1, 1\rangle$, $\langle 6, 39, 23, 2, 2\rangle$, $\langle 6, 42, 25, 1, 1\rangle$,
$\langle 7, 44, 26, 1, 1\rangle$, $\langle 8, 30, 18, 2, 2\rangle$, $\langle 8, 46, 27, 1, 1\rangle$, $\langle 9, 42, 25, 1, 1\rangle$, $\langle 10, 39, 23, 2, 2\rangle$,
... ...
$\langle 27, 30, 18, 1, 2\rangle$, $\langle 28, 32, 19, 1, 2\rangle$, $\langle 28, 41, 24, 2, 1\rangle$, $\langle 29, 29, 17, 1, 2\rangle$, $\langle 30, 27, 16, 1, 2\rangle$,
$\langle 30, 50, 29, 2, 1\rangle$, $\langle 31, 29, 17, 1, 2\rangle$ }



7. Representing music using multidimensional datasets (1)

1. Most of the short-comings in these algorithms arise from the fact that they're based on the idea of representing a piece of music as a one-dimensional string of symbols or, in the case of polyphonic music, a set of such strings.

2. It turns out that you can overcome all of these problems and end up with an algorithm that is capable of discovering all of the different types of interesting repeated pattern that I mentioned earlier simply by abandoning the string-based approach in favour of one where the music is represented as a multidimensional dataset.

3. A multidimensional dataset is actually any set of points in a space with any number of dimensions. The algorithms that we're going to describe work with datasets of any dimensionality and any size. Also the co-ordinates can take real values (which, of course, in an implementation would be represented as floating point values).

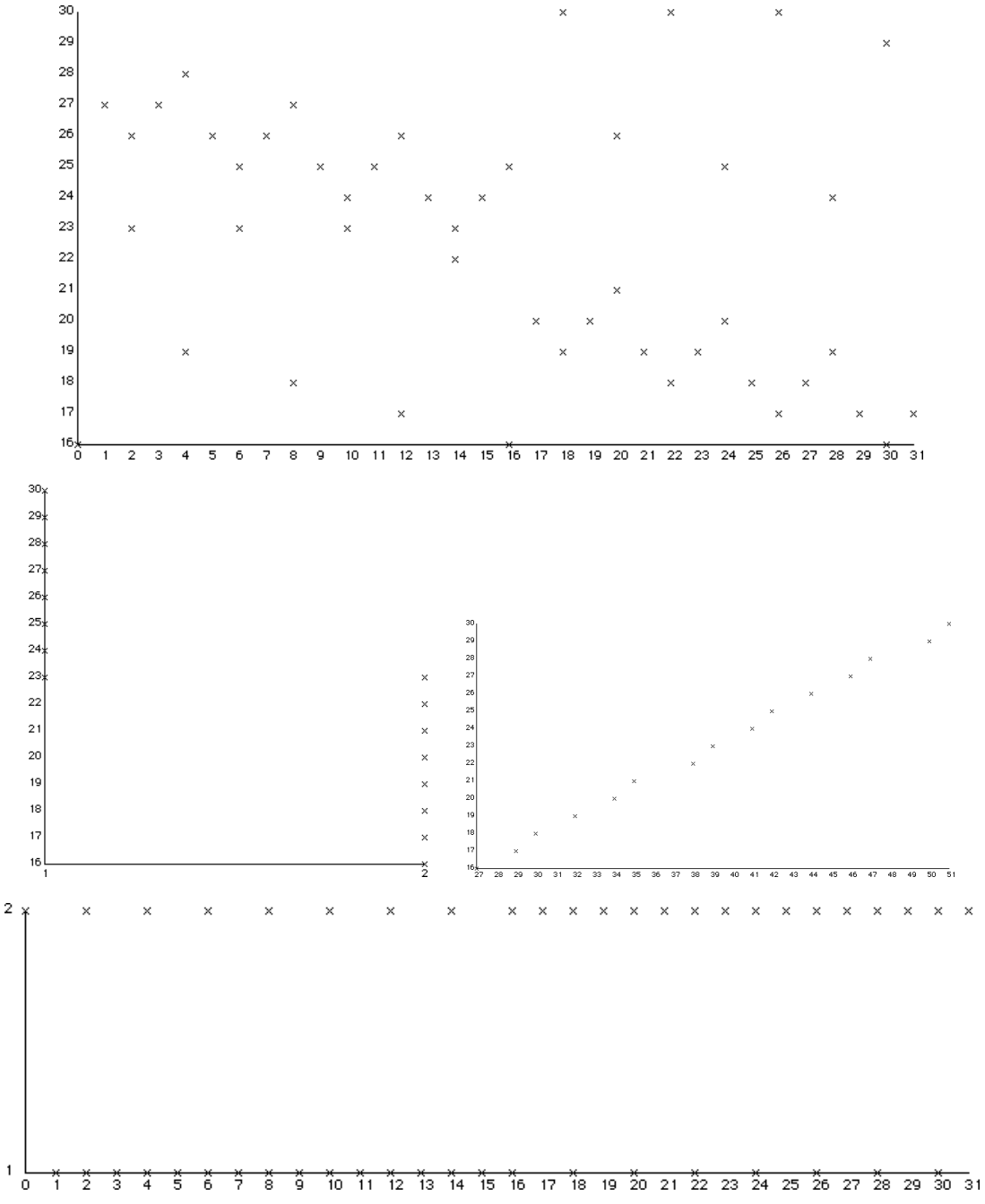
4. There are many possible appropriate ways of representing a piece of music as a multidimensional dataset and this example here shows some of the simpler possibilities.

5. At the top here we have the first two bars of the second *Prelude* from book 2 of Bach's *Wohltemperirte Klavier*.

6. Then underneath we have a 5-dimensional dataset that represents of this score. The co-ordinate values in each datapoint represent onset time, chromatic pitch, morphetic pitch (which is continuous diatonic pitch), duration and voice. Each datapoint represents a single note event.

7. What we can then do is take various different types of orthogonal projection of such a dataset. For example, we could just consider the first two dimensions and get this 2-dimensional projection which tells us the chromatic pitch and onset time of each note.

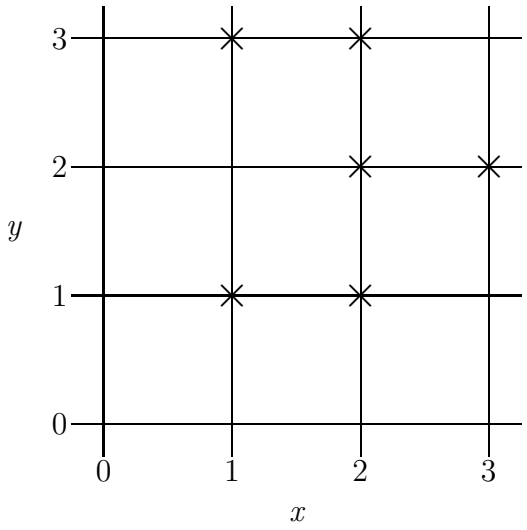
8. Representing music using multidimensional datasets (2)



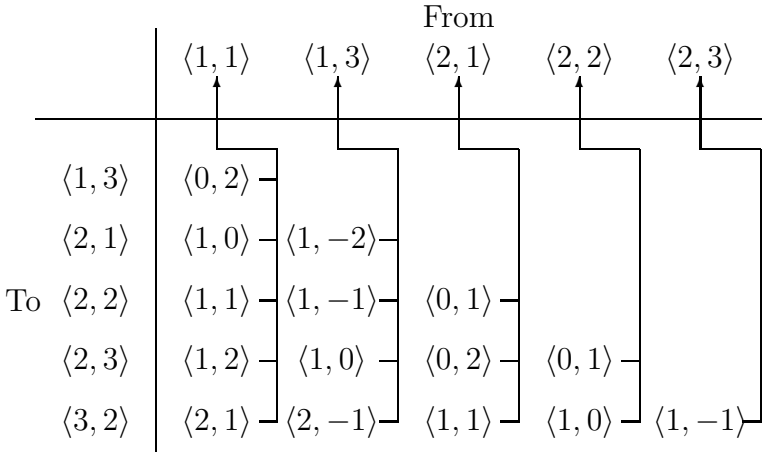
8. Representing music using multidimensional datasets (2)

1. A number of the other possible two-dimensional projections of this dataset also give us useful information.
2. For example, this first one is a graph of morphetic pitch against onset time. Note that some of the patterns that were only similar in the chromatic pitch against onset time graph are now identical because we're using a representation of diatonic pitch. It's often more profitable when analysing tonal music to look for exact repetitions in this type of projection than in the chromatic pitch representation.
3. Here's another projection which shows pitch against voice and shows the range of each voice rather nicely.
4. This projection shows morphetic pitch against chromatic pitch and gives a representation of the pitch set that's used in the passage. In this particular case it shows quite clearly that the passage is in G major.
5. Finally, this projection shows duration against onset time.

9. SIA: Discovering maximal repeated patterns in multidimensional datasets



Vector		Datapoint
$\langle 0,1\rangle$	$\rightarrow$	$\boxed{\langle 2,1\rangle}$
$\langle 0,1\rangle$	$\rightarrow$	$\boxed{\langle 2,2\rangle}$
$\langle 0,2\rangle$	$\rightarrow$	$\boxed{\langle 1,1\rangle}$
$\langle 0,2\rangle$	$\rightarrow$	$\boxed{\langle 2,1\rangle}$
$\langle 1,-2\rangle$	$\rightarrow$	$\boxed{\langle 1,3\rangle}$
$\langle 1,-1\rangle$	$\rightarrow$	$\boxed{\langle 1,3\rangle}$
$\langle 1,-1\rangle$	$\rightarrow$	$\boxed{\langle 2,3\rangle}$
$\langle 1,0\rangle$	$\rightarrow$	$\boxed{\langle 1,1\rangle}$
$\langle 1,0\rangle$	$\rightarrow$	$\boxed{\langle 1,3\rangle}$
$\langle 1,0\rangle$	$\rightarrow$	$\boxed{\langle 2,2\rangle}$
$\langle 1,1\rangle$	$\rightarrow$	$\boxed{\langle 1,1\rangle}$
$\langle 1,1\rangle$	$\rightarrow$	$\boxed{\langle 2,1\rangle}$
$\langle 1,2\rangle$	$\rightarrow$	$\boxed{\langle 1,1\rangle}$
$\langle 2,-1\rangle$	$\rightarrow$	$\boxed{\langle 1,3\rangle}$
$\langle 2,1\rangle$	$\rightarrow$	$\boxed{\langle 1,1\rangle}$

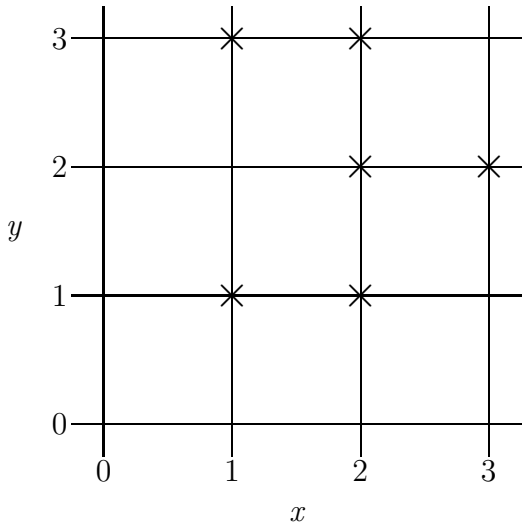


9. `SIA`: Discovering maximal repeated patterns in multidimensional datasets

1. `SIA` takes a multidimensional dataset as input and finds for every possible vector v the largest pattern in the dataset that can be translated by v to give another pattern in the dataset.
2. For example, if we consider this dataset here, then the largest pattern that can be translated by the vector $\langle 1, 0\rangle$ is the pattern that consists of these three points $\{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 1, 3\rangle\}$.
3. And the largest pattern that can be translated by the vector $\langle 1, 1\rangle$ is the pattern that consists of these two points $\{\langle 1, 1\rangle, \langle 2, 1\rangle\}$.
4. We say that a pattern is *translatable* by a given vector if it can be translated by the vector to give another pattern that is a subset of the dataset.
5. The *maximal translatable pattern* for a given vector is then the largest pattern that can be translated by the vector to give another pattern that is in the dataset.
6. `SIA` discovers all the non-empty maximal translatable patterns for a given dataset and it does it like this:
7. First it constructs this table here which we call the *vector table* for the dataset. A cell in the table contains the vector *from* the datapoint at the head of the column of that cell *to* the datapoint at the head of the row for that cell. [GIVE EXAMPLE].
8. `SIA` computes all the values in this table below the leading diagonal as shown here. In other words, it computes for each datapoint all the vectors from that datapoint to every other datapoint in the dataset greater than it.
9. Note that each of these vectors is stored with a pointer that points back to the "origin" datapoint for which it was computed (that is, the datapoint at the top of its column).
10. Actually, before computing this table, the dataset is sorted using merge sort. This means that the vectors increase as you descend the column and decrease as you move from left to right across a row.
11. Having constructed this table, `SIA` then simply sorts the vectors in the table using a slightly modified version of merge sort to give a list like this one here on the right-hand side.
12. Note that each vector in this list is still linked to the datapoint at the head of its column in the vector table. Simply reading off all the datapoints attached to the adjacent occurrences of a given vector in this list therefore yields the maximal translatable pattern for that vector.
13. The complete set of non-empty maximal translatable patterns can be obtained simply by scanning the list once, reading off the attached datapoints and starting a new pattern each time the vector changes. Each box in the right-hand column of the list corresponds to a maximal translatable pattern.

14. The most expensive step in this process is sorting the vectors which can be done in a worst-case running time of $O(kn^2 \log_2 n)$ for a k-dimensional dataset of size n.

10. SIATEC: Discovering all the occurrences for each maximal translatable pattern



		From					
		$\langle 1,1\rangle$	$\langle 1,3\rangle$	$\langle 2,1\rangle$	$\langle 2,2\rangle$	$\langle 2,3\rangle$	$\langle 3,2\rangle$
	$\langle 1,1\rangle$	$\langle 0,0\rangle$	$\langle 0,-2\rangle$	$\langle -1,0\rangle$	$\langle -1,-1\rangle$	$\langle -1,-2\rangle$	$\langle -2,-1\rangle$
	$\langle 1,3\rangle$	$\langle 0,2\rangle$	$\langle 0,0\rangle$	$\langle -1,2\rangle$	$\langle -1,1\rangle$	$\langle -1,0\rangle$	$\langle -2,1\rangle$
	$\langle 2,1\rangle$	$\langle 1,0\rangle$	$\langle 1,-2\rangle$	$\langle 0,0\rangle$	$\langle 0,-1\rangle$	$\langle 0,-2\rangle$	$\langle -1,-1\rangle$
To	$\langle 2,2\rangle$	$\langle 1,1\rangle$	$\langle 1,-1\rangle$	$\langle 0,1\rangle$	$\langle 0,0\rangle$	$\langle 0,1\rangle$	$\langle -1,0\rangle$
	$\langle 2,3\rangle$	$\langle 1,2\rangle$	$\langle 1,0\rangle$	$\langle 0,2\rangle$	$\langle 0,1\rangle$	$\langle 0,0\rangle$	$\langle -1,1\rangle$
	$\langle 3,2\rangle$	$\langle 2,1\rangle$	$\langle 2,-1\rangle$	$\langle 1,1\rangle$	$\langle 1,0\rangle$	$\langle 1,-1\rangle$	$\langle 0,0\rangle$

Time to find all occurrences of pattern $p = O(|p|n)$.

$$\sum_{i=1}^{l} |p_i| \leq \frac{n(n-1)}{2}$$

$$O\left(\sum_{i=1}^{l} |p_i| n\right) \leq O\left(\frac{n^2(n-1)}{2}\right)$$

Overall worst-case running time of SIATEC $= O(kn^3)$

10. SIATEC: Discovering all the occurrences for each maximal translatable pattern

1. SIATEC first generates all the maximal translatable patterns using a slightly modified version of SIA and then it finds all the occurrences of each of these patterns.
2. I explained on the previous slide that SIA only computes the vectors below the leading diagonal in the vector table. This is because the maximal translatable pattern for a vector $-v$ is the same as the pattern that you get by translating the maximal translatable pattern for v by the vector v itself. [DEMONSTRATE ON SLIDE].
3. However, it turns out that by computing *all* the vectors in the vector table we can more efficiently discover all the occurrences of any given pattern within the dataset.
4. So in SIATEC we actually compute this complete table here and we use the region below the leading diagonal to compute the maximal translatable patterns as in SIA.
5. We sort the dataset before computing the table so that the vectors increase as you descend a column and decrease as you move from left to right along a row.
6. Now, we know that a given column contains all the vectors that the datapoint at the top of the column can be translated by to give another point in the dataset.
7. Say we want to find all the occurrences of the pattern $\{\langle 1,1\rangle, \langle 2,1\rangle\}$ which is the maximal translatable pattern in this dataset for the vector $\langle 1,1\rangle$.
8. Now, when we say that we want to "find all the occurrences" of a pattern, all we actually need to find is the set that contains all and only those vectors that we can translate the pattern by to get another pattern that is in the dataset.
9. We know that the column of vectors under $\langle 1,1\rangle$ contains all the vectors that this datapoint can be translated by and we know that the column under $\langle 2,1\rangle$ contains all the vectors that this datapoint can be translated by. So a pattern is only translatable by the vectors in the intersection of the columns in the vector table corresponding to the points in the pattern.
10. So, for example, the pattern $\{\langle 1,1\rangle, \langle 2,1\rangle\}$ can only be translated by every vector that is in both the column headed by $\langle 1,1\rangle$ *and* the column headed by $\langle 2,1\rangle$.
11. In other words, to find the set of occurrences for a given pattern we simply have to find the intersection of the columns headed by the datapoints in that pattern.
12. By exploiting the orderedness of this table, it's possible to find all the occurrences of a pattern p in a dataset of size n in a worst-case running time of $O(|p|n)$ where $|p|$ is the number of datapoints in the pattern p.
13. We know that the complete set of maximal translatable patterns is found by SIA simply by sorting the vectors below the leading diagonal in this vector table. If there

are l such patterns then this implies

$$\sum_{i=1}^{l} |p_i| \leq \frac{n(n-1)}{2}$$

where $|p_i|$ is the cardinality of the ith pattern.

14. The overall worst-case running time of `SIATEC` is therefore

$$O\left(\sum_{i=1}^{l} |p_i| n\right) \leq O\left(\frac{n^2(n-1)}{2}\right)$$

Therefore the algorithm is $O(n^3)$ (or, in fact, $O(kn^3)$ for a k-dimensional dataset).

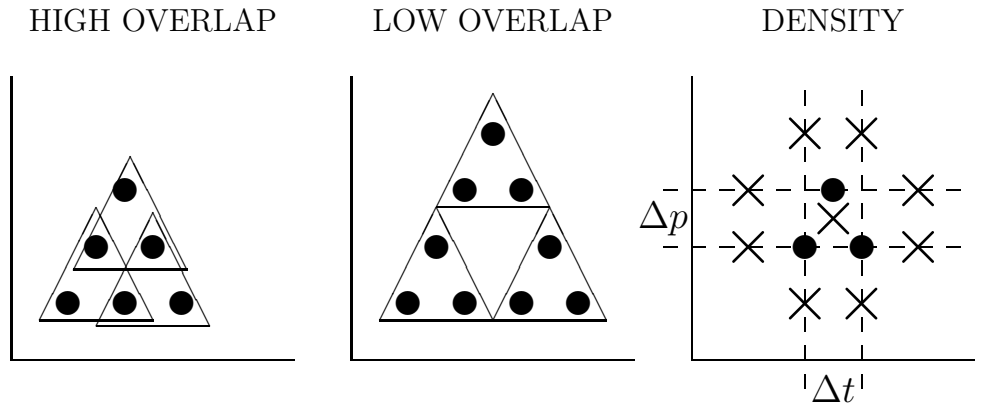
11. MU: Selecting the musically interesting repeated patterns

- Number of patterns in a dataset of size $n = 2^n$
- Number of patterns generated by SIA $< \frac{n^2}{2}$
- Experiments suggest that most of the interesting patterns are included among the the patterns generated by SIA.
- BUT many of the patterns generated by SIA are *not* musically interesting.
  - Over 70000 patterns discovered for Rachmaninoff *Prelude* Op.3 No.2.
  - Far fewer than 1000 of these are going to be analytically interesting.
  - Maybe a larger proportion of the patterns could be used by a data compression application.
  - So typically less than 1% of the patterns generated by SIA would be regarded as musically significant by an analyst or expert listener.
- MU: a system that evaluates the output of SIATEC and isolates the musically interesting repeated patterns.

11. MU: Selecting the musically interesting repeated patterns

1. A dataset of size n contains 2^n distinct subsets.
2. The number of patterns generated by SIA is less than $\frac{n^2}{2}$.
3. SIA discovers all the maximal translatable patterns in the powerset of a dataset and typically this set of maximal translatable patterns is only a tiny proportion of the patterns in the powerset of the dataset.
4. Our experiments seem to indicate that the set of patterns generated by SIA typically contains many of the musically interesting repeated patterns that we want to find, which is very good.
5. Nevertheless, only a very small proportion of the patterns generated by SIA would be considered musically interesting by an analyst or expert listener. [SEE EXAMPLE ON SLIDE].
6. Although a much larger proportion of the generated patterns might be used in a data compression application.
7. So we can say that typically less than 1% of the patterns generated by SIA for a reasonably-sized piece of music would be regarded as musically interesting by an analyst or or expert listener.
8. This means that we need to devise a system that evaluates the output of SIATEC and isolates the musically interesting repeated patterns.
9. I am currently developing a system that I call MU which does just this.

12. `MU`: Some possible heuristics



OVERLAP $= \frac{3\times3}{6} = 1.5$ OVERLAP $= \frac{3\times3}{9} = 1$ TEMP. DENS $= \frac{3}{\Delta t}$

NOTE DENS. (TIME) = 3/8

NOTE DENS. (TIME & PITCH) = 3/4

Possible heuristics for finding "theme-like" patterns:

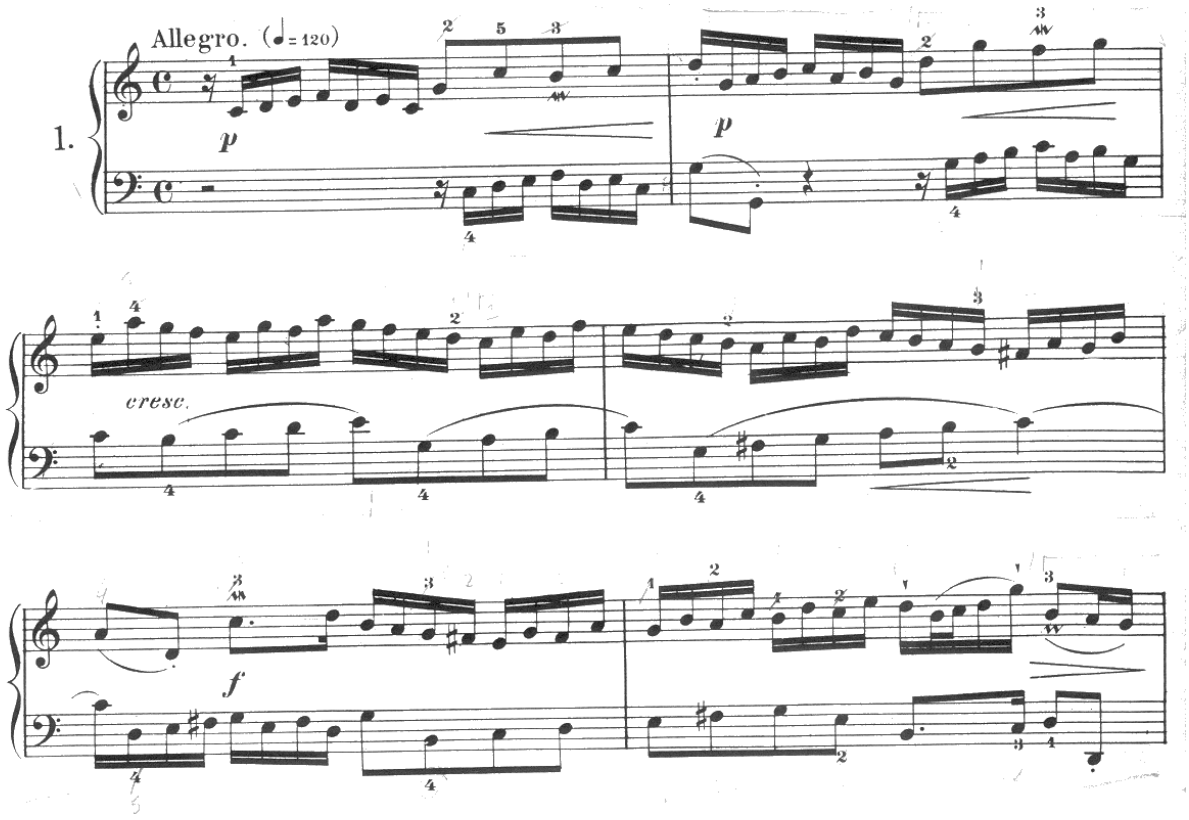
1. Frequency of occurrence.
2. Size of pattern.
3. Overlap.
4. Density
   (a) Temporal density.
   (b) Note density.
      i. Region defined as time interval spanned by pattern.
      ii. Region defined as time interval and pitch range spanned by pattern.

12. `MU`: Some possible heuristics

1. We therefore come back to the problem that I mentioned earlier which is that of formally characterising the class of "interesting" repetitions.
2. Unfortunately, it seems that there are many different ways in which a repeated musical structure can be "interesting".
3. Or, to put it another way, it seems that there are many different types of interesting fact that a repeated pattern might be able to tell us something about.
4. For example, the largest repeated pattern in the *Prelude* in C minor from Book 2 of Bach's 48 (the one that I showed you the first two bars of earlier on) tells us that there are 115 occasions in the piece when a note is followed precisely one bar later by a note that is a major second lower.
5. This reflects the fact that there is a great deal of sequence at a bar's interval within the piece.
6. Then, of course, there are the more theme-like repeated patterns which consist of a set of notes that are more-or-less contiguous in the music.
7. My main point here is that there is probably no single set of consistent heuristics that will isolate all and only those repeated patterns that are musically interesting.
8. So we need a number of different sets of heuristics, each set tailored to isolating patterns of a particular type.
9. At the moment I'm trying to devise a set of heuristics that will isolate "theme-like" musical patterns—that is, the type of thing you might find in a musical thematic index.
10. What `MU` actually does at the moment is compute for each pattern p a numerical value—let's call it $Q(p)$—that is intended to reflect how "theme-like" the pattern is.
11. This is a list of some of the heuristics that I'm experimenting with at the moment:
    (a) *FREQUENCY OF OCCURRENCE* The more frequently a pattern is repeated, the better.
    (b) *PATTERN SIZE* The larger a pattern, the better.
    (c) *OVERLAP* The fewer the number of notes shared between separate occurrences of the pattern, the better. This reflects the intuition that for a pattern to perceived as being an individual unit, it must not share notes with repetitions of the pattern.
    (d) *DENSITY* The denser the pattern, the better. There are a number of ways of measuring the density of a pattern:
        i. *Temporal density* Divide the number of notes in the pattern by the time interval spanned by the pattern.

ii. *Note density* Divide the number of notes in the pattern by the number of notes in region of the piece spanned by the pattern. There are a number of possible ways to define the "region spanned by a pattern". For example, this could be the set of all notes that occur during the time interval spanned by the pattern. Or it could be all the notes that occur within the time interval *and* pitch range spanned by the pattern and so on.

13. Some preliminary results



13. Some preliminary results

1. I've been focusing recently on finalising `SIATEC` and I've only just started working again on `MU`. So I haven't been able yet to run `MUSIATEC` on any full-blown examples.
2. What I can show you, however, is the result of running an early `MUSIATEC` prototype on some fairly small musical examples of about 1-200 notes.
3. So, in this first example, we've got the first 109 notes (or 5 bars) of Bach's Two-part Invention in C major. `SIATEC` generates 857 maximal translatable patterns for this passage and one of the ones that `MU` computes to be the most "theme-like" is this pattern of seven notes shown here [SEE SLIDE].
4. Here's what the passage sounds like with the occurrences of this pattern emphasized. [PLAY invention-c.mid].
5. [Put on slide 5].
6. When I ran the prototype `MUSIATEC` on this passage from the beginning of Barber's Piano Sonata, this repeated bass figure is amongst those patterns that are evaluated to be the most "theme-like".
7. [PUT ON SLIDE 4]
8. Similarly, when I run the program on the first 6 bars of the Rachmaninoff Prelude in C sharp minor this descending motif is one of the patterns given the highest score by `MU`.

14. Some possible applications of SIA and SIATEC

- Extraction of similar patterns in a computational model of music cognition.
- COMPRESSION
  - Music
  - Video
  - Databases of 3-d molecular structures
  - Graphics
  - ...
- DATABASE INDEXING
  - Music
  - Video
  - 3-d molecular structures
- ANALYSIS ('DATA-MINING')
  - Music
  - Financial and scientific experimental data
  - 3-d molecular structures

14. Some possible applications of SIA and SIATEC

1. Clearly, apart from potentially forming a fundamental component in a computational model of expert music cognition, SIA and SIATEC could also be used as the basis of a number of practical applications such as these listed here.
2. I'll now hand you over to Geraint who will describe his SIA(M)ESE pattern-matching algorithm.

Bibliography

I. Bent and W. Drabkin. *Analysis*. New Grove Handbooks in Music. Macmillan, 1987.

E. Cambouropoulos. *Towards a General Computational Theory of Musical Structure*. PhD thesis, University of Edinburgh, Feb. 1998.

J.-L. Hsu, C.-C. Liu, and A. L. Chen. Efficient repeating pattern finding in music databases. In *Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management*, pages 281–288. Association of Computing Machinery, 1998.

F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. M.I.T. Press, Cambridge, Mass., 1983.

J.-J. Nattiez. *Fondements d'une sémiologie de la musique*. Union Générale d'Éditions, Paris, 1975.

P.-Y. Rolland. Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4):334–350, 1999.

G. Rouget. Un chromaticisme africain. *l'Homme*, I(3), 1961.

N. Ruwet. Méthodes d'analyse en musicologie. *Revue belge de musicologie*, 20:65ff., 1966. Republished in (Ruwet, 1972, 100–134); English translation in *Music Analysis*, Vol.5, 1986.

N. Ruwet. *Langage, Musique, Poésie*. Éditions du seuil, 27, rue Jacob, Paris VI., 1972.