Extracting 'Significant' Patterns from Musical Strings: Some Interesting Problems.

Emilios Cambouropoulos

Austrian Research Institute for Artificial Intelligence
Vienna, Austria
emilios@ai.univie.ac.at

Abstract

In this paper a number of issues relating to the application of string processing techniques on musical sequences are discussed. Special attention is given to musical pattern extraction. Firstly, a number of general problems are presented in terms of musical representation and pattern processing methodologies. Then a number of interesting melodic pattern matching problems are presented. Finally, issues relating to pattern extraction are discussed, with special attention being drawn to defining musical pattern 'significance'. This paper is not intended towards providing solutions to string processing problems but rather towards raising awareness of primarily music-related particularities that can cause problems in matching applications and also suggesting some interesting string processing problems that require efficient computational solutions.

1. Introduction

It is often hypothesised that a musical surface may be seen as a string of musical entities such as notes, chords etc. on which pattern recognition or induction techniques can be applied. In this text, the term *pattern induction or extraction* refers to techniques that enable the extraction of useful patterns from a string whereas *pattern recognition* refers to techniques that enable locating all the instances of a predefined pattern in a given string. Overviews of the application of pattern processing algorithms on musical strings can be found in (McGettrick, 1997; Crawford et al, 1998; Rolland et al, 1999).

2. Issues of Musical Pattern Representation

2.1. Pattern Matching vs Pattern Extraction (Problem of Significance)

One of the differences between pattern matching and pattern induction techniques is that the latter requires a notion of pattern 'significance'. Pattern matching techniques do not encounter this problem because the search query is given; the user has decided *a priori* that a certain pattern is important and then all the matches in a string or set of strings are located. In pattern extraction, however, one has to decide *what* types of patterns the algorithm should look for - finding *all* the patterns is often not very useful. Selecting 'significant' patterns can be done either *after* all the patterns have been found (which is not usually the most efficient approach) or *before* by forcing algorithms to stop when the specific types of patterns are found (e.g. periods or covers). The former approach is briefly discussed in section 4.2 whereas the latter in section 4.3.

Presented at the *London String Days 2000* workshop, 3-4 April 2000, King's College London and City University.

2.2 Musical Notes vs Musical Relations between Notes

Expressive MIDI files are adequate for searching pitch patterns but are problematic in terms of rhythm patterns. The reason is that MIDI data are not quantised, i.e. onsets, durations and interonset intervals are not categorically organised so they can not be represented by the usual symbolic nominal musical values (e.g. quarter notes etc). MIDI files require preprocessing so that they can be converted to a score-like format - one computational system for score extraction from MIDI files is presented in (Cambouropoulos, 2000). However, the algorithms discussed in section 3 can be used for approximate matching on melodies in the time domain, in which case quantisation may not be necessary.

A melodic sequence is commonly represented as a set of independent strings of elementary musical parameters, e.g. pitch and duration, or alternatively as strings of relations between adjacent notes, e.g. pitch intervals and duration ratios.

In the pitch domain, the main problem with applying a pattern-processing algorithm on an absolute pitch string is that transpositions are not accounted for. There is plenty of evidence, both theoretical and experimental, that transposition is paramount in the understanding of musical patterns. The obvious solution to this problem is the use of relative pitch, mainly through the derivation of pitch intervals from the absolute pitch surface. It is herein maintained that pattern-matching and pattern-induction algorithms should be developed primarily for sequences of pitch intervals. As will be shown in section 4.3, pattern induction algorithms that can be applied on absolute pitch sequences may not be meaningful for pitch interval sequences. An extended discussion on pitch representation for pattern matching can be found in (Cambouropoulos et al, 2000).

In terms of the rhythmic component of musical strings, string-processing algorithms are most commonly applied to strings of durations or inter-onset intervals. This type of matching can be very effective, but one should also consider encoding rhythm strings as strings of duration relations such as duration ratios or shorter/longer/equal strings. Duration ratios encapsulate the observation that listeners usually remember a rhythmic pattern as a relative sequence of durations that is independent of an absolute tempo. Duration ratios can reveal, for instance, augmentations or diminutions of a rhythmic pattern.

2.3 1-D vs 2-D Matching

A polyphonic musical work can be represented either as a 2-dimensional graph (pitch against time) or as a collection of 1-dimensional strings. In the former case, special algorithms have to be used for finding patterns in a two dimensional space. Such algorithms are very useful because most commonly musical databases contain simple unstructured MIDI files. Additionally they enable the retrieval of polyphonic structures rather than just melodic patterns (see Dovey 1999). One potential problem is that, if a (melodic) search query is not long enough and also contains large pitch leaps, any algorithm is likely to return a large number of instances that are musically and/or perceptually implausible.

The second representation requires sophisticated streaming algorithms, i.e. algorithms that can split the polyphonic work into ‘meaningful’ independent streams (or voice parts). This is not a trivial task. The development however of such algorithms can be very useful for preparing the musical data for pattern processing tasks.

A preliminary version of such an algorithm is presented in (Cambouropoulos, 2000). The streaming algorithm is based on the Gestalt principle of proximity and simply tries to find the shortest streams that connect all the onsets within a beat (figure 1). Crossing of streams is not allowed. The number of streams is always equal to the number of notes in the largest chord. The solution to this problem is not trivial and appropriate searching techniques are required for developing an efficient algorithm. The current elementary version of the algorithm makes mistakes (see figure 2) but can be improved if other principles like ‘goodness of continuation’ are taken into account. Streaming is a large research topic in its own right (see Bregman 1990).

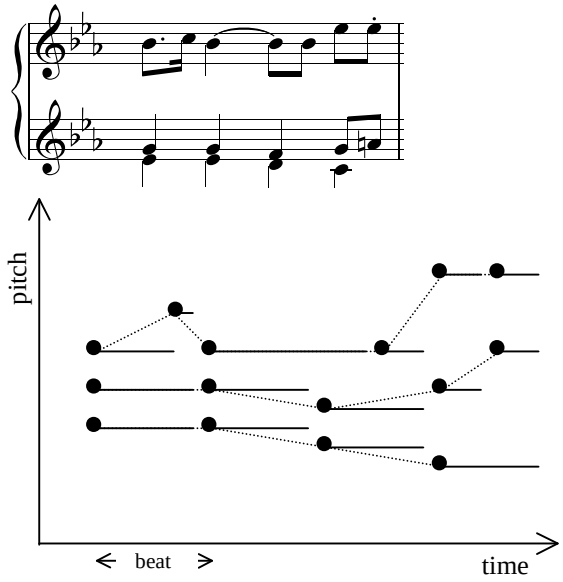


Figure 1 Application of streaming algorithm on beginning of Mozart's *Sonata KV282*. Dots in the graph represent the onsets of the notes in the musical segment; dotted lines show the three streams detected by the streaming algorithm; horizontal lines indicate the inter-onset intervals for each stream.

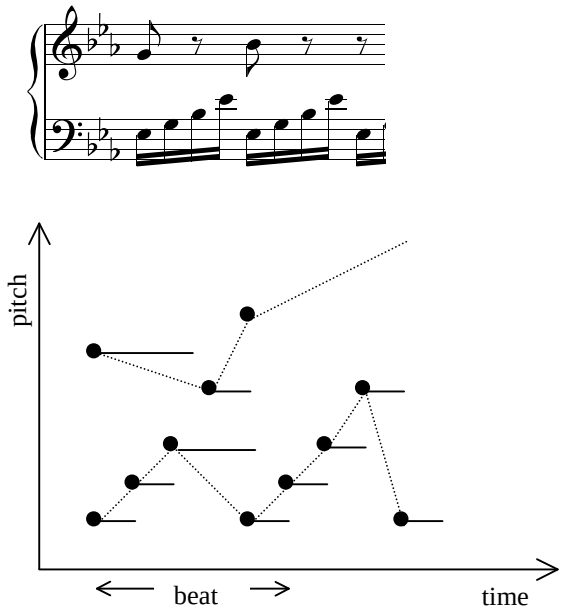


Figure 2 The streaming algorithm fails locally on this excerpt from Mozart's *Sonata KV282* (see caption of figure 1 for explanation of graph).

3. Pattern Matching

In this section a number of interesting pattern matching problems for strings consisting of integers will be presented. These involve primarily matching problems in the pitch domain but some could also be extended in the time domain.

3.1 Patterns with Similar Intervals

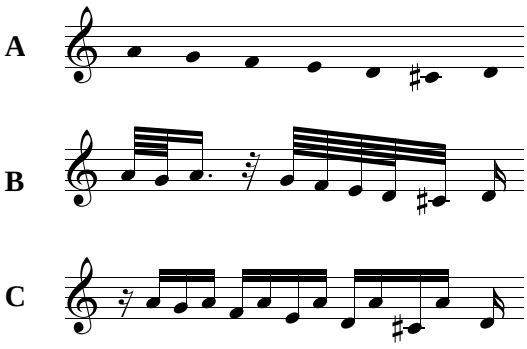
Most computer-aided musical applications adopt an absolute numeric pitch representation - most commonly MIDI pitch and pitch intervals in semitones; duration is also encoded in a numeric form. In all the examples below melodic strings are represented as strings of *pitch intervals in semitones*.

One way to account for similarity between closely related but non-identical musical strings is to use what will be referred to as *δ-approximate matching*. In δ-approximate matching, equal-length patterns consisting of integers match if each pair of corresponding integers differ by not more than δ - e.g. an ascending major chord arpeggio [+4, +3, +5] and a minor arpeggio [+3, +4, +5] sequence can be matched if a tolerance δ=1 is allowed in the matching process (the total sum of δ tolerance allowed for a pattern match can be constrained by a further γ tolerance parameter resulting in *δ-γ approximate matching*). Efficient algorithms for solving these problems are presented in (Cambouropoulos et al, 1999).

3.2 Filling and Thinning of Patterns

The above algorithm for δ-approximate matching accounts only for equal length patterns. A common technique of musical composition is *filling* and *thinning* of musical motivic and thematic material. That is, extra notes are added in a musical pattern (filling) or taken away (thinning). Approximate matching algorithms that can account for this phenomenon rely usually on dynamic programming techniques. In this section we will merely try to describe in more detail this problem. The melodic examples presented in this section are taken from the classical study on thematic processes by Reti (1951).

Adding a note between two notes essentially can be interpreted as splitting the initial pitch interval into two successive intervals the sum of which is equal to the initial interval - e.g. initial sequence 60, 62 (interval: +2); sequence with added note: 60, 67, 62 (intervals: +7, -5); the sum of the two resulting intervals is equal to the initial interval. This property can be used for matching different length sequences by allowing one interval of one string to be matched against two or more successive intervals of the other string whose sum is equal (or δ-approximate) to the initial interval. See Figures 3-5.



A -2 -2 -1 -2 -1 1
B -2 2 -2 -2 -1 -2 -1 1

A -2 -2 -1 -2 -1 1
C -2 2 -4 +4 -5 -5 -7 7 -8 8 -7

B -2 2 -2 -2 -1 -2 -1 1
C -2 2 -4 +4 -5 -5 -7 7 -8 8 -7

Figure 3 Beginning of Toccata (B) and theme of Fugue (C) from Bach's *D-minor Toccata and Fugue*

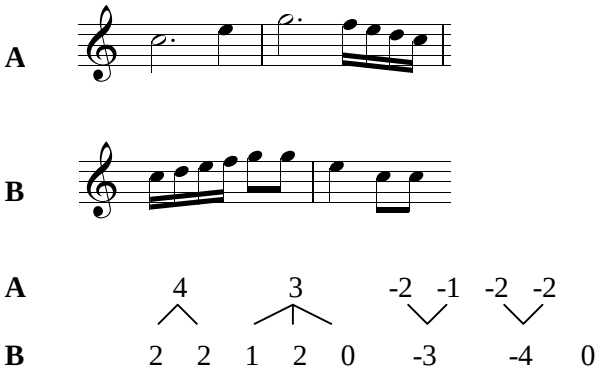


Figure 4 First Allegro theme (A) and first Finale theme (B) from Beethoven's *First Symphony* (pattern B is also the retrograde of pattern A).

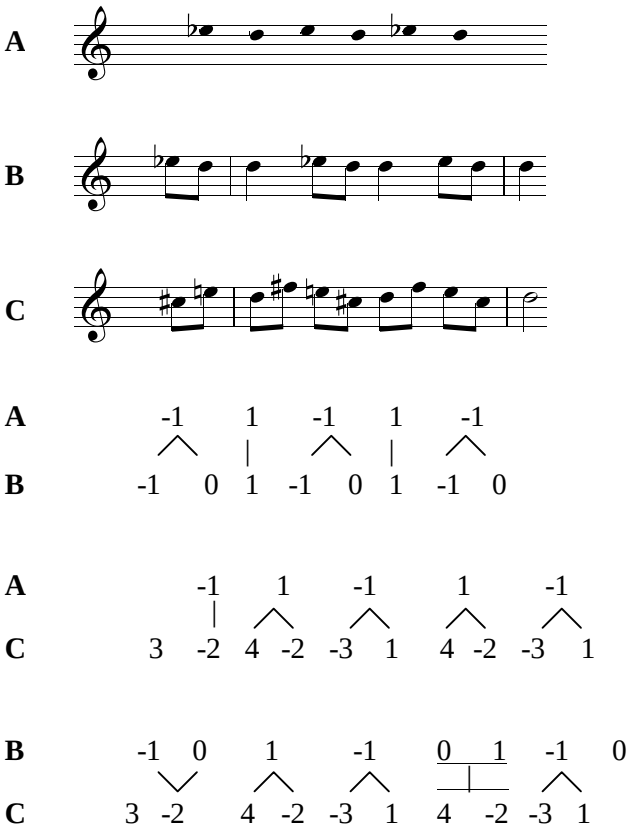
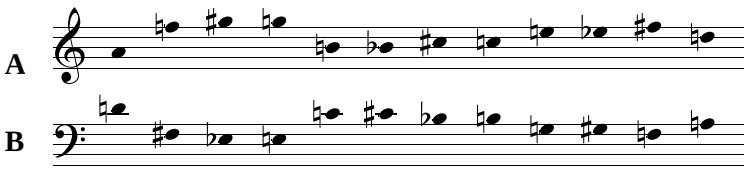


Figure 5 Opening theme (B) and part of Finale theme (C) of Mozart's *Symphony in G minor*

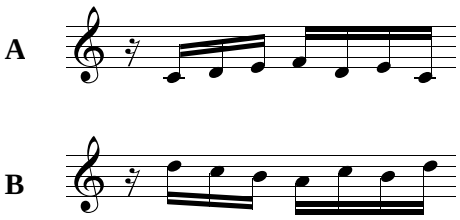
3.3 Retrogrades and Inversions

Inversions of patterns can be matched if the absolute of the *sum* of the corresponding intervals of the original and the inversion is not more than δ. See figures 6 and 7.



A	8	3	-1	1	-8	-1	3	-1	4	-1	3	-4
B	-8	-3	1	-1	8	1	-3	1	-4	1	-3	4
|Sum|	0	0	0	0	0	0	0	0	0	0	0	0

Figure 6 Original and Inversion of 12-tone series in Webern's *Cantata No.1, Op.29*



A	2	2	1	-3	2	-3
B	-2	-1	-2	3	-1	3
|Sum|	0	1	1	0	1	0

Figure 7 Two instances of a motive from Bach's *Two Part Inventions*, No.1 (BWV 772) – δ=1.

It would be very useful to have *one* algorithm that can do all the above types of matching presented in these sections (3.1, 3.2 and 3.3) by allowing control of different parameters.

4. Pattern Extraction

4.1 Finding All Patterns

An efficient algorithm that computes *all* the exact repetitions in a given string is described in (Crochemore, 1981; Iliopoulos et al., 1996). For a given string of symbols (e.g. string of pitch intervals), the matching process starts with the smallest pattern length and ends when the largest pattern match is found. This algorithm takes $O(n \cdot logn)$ time where n is the length of the string. Dynamic programming algorithms can be used for finding *all* the approximate repetitions in a string.

It is apparent that such a procedure for the discovery of all identical melodic patterns (even more so for approximate matching) will produce an extremely large number of possible patterns most of which would be considered counter-intuitive and non-pertinent by a human musician/analyst. So the problem of pattern 'significance' arises.

4.2 Pattern Significance (a posteriori)

Firstly, pattern significance can be determined *after* all the patterns have been found. According to one such procedure proposed in (Cambouropoulos 1998) a prominence value is attached to each of the discovered patterns based on the following factors: a) prefer longer patterns, b) prefer most frequently

occurring patterns, c) avoid overlapping. A *selection function* that calculates a numerical strength value for a single pattern according to the these principles can be devised, for instance:

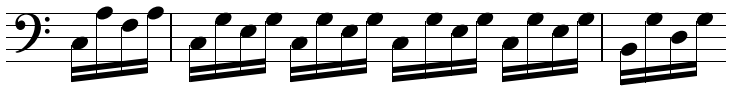
$f(L,F,DOL)=F^{a}\cdot L^{b}/10^{c\cdot DOL}$

where: *L:* pattern length; *F:* frequency of occurrence for one pattern; *DOL:* degree of overlapping; *a, b, c:* constants that give different prominence to the above principles. For every pattern discovered by the above exact pattern induction algorithm a value is calculated by the selection function. The patterns that score the highest should be the most significant ones.

4.3 Pattern Significance (a priori)

An alternative approach, is determining types of significant patterns *in advance* so as to enable algorithms to stop as soon as the appropriate significance criteria are met. 'Significant' types of patterns are, for instance, squares, periods and covers; for example, *abc* is a period of *abcabcabca*, and *abca* is a cover of *abcabcaabca* (these specific types of patterns are important in biological string processing applications).

What types of patterns are 'significant' for musical extraction tasks? One possibly interesting type of musical pattern may relate to immediate repetitions (2 or more consecutive repetitions). The obvious type of pattern that would seem appropriate for finding such consecutive repetitions is the *period.* This is true for the absolute pitch domain (which is not very interesting) and for the inter-onset interval domain (which is very useful). For the pitch interval domain (and interonset interval ratio domain), however, some other type of pattern is necessary for finding immediate repetitions. We will call this type of pattern a *disjunct period* which is essentially a repeating pattern separated by single symbols. For example, *abc* is a disjunct period of *abcdabcaabcbabc*. These separating symbols (intervals) are necessary if consecutive pitch patterns are expected not to overlap. See figures 8 & 9.

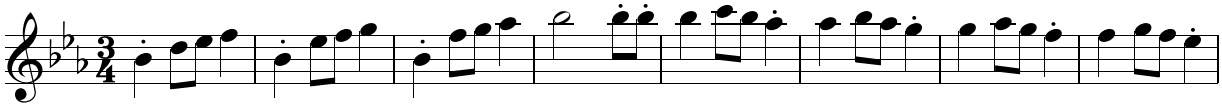


9 -4 4 -9 7 -3 3 -7 7 -3 3 -7 7 -3 3 -7 7 -3 3 -8 8 -5 5 local disjunct period

9 -4 4 -9 7 -3 3 -7 7 -3 3 -7 7 -3 3 -7 7 -3 3 -8 8 -5 5 local period

9 -4 4 -9 7 -3 3 -7 7 -3 3 -7 7 -3 3 -7 7 -3 3 -8 8 -5 5 local cover

Figure 8 Section from Alberti Bass



	pci	4	1	2	-7	5	2	2	-9	7	2	1	2	0	0	0	2	-2	-2	0	2	-2	-1	0	1	-1	-2	0	2	-2	-2
	nci	2	1	1	-5	3	1	1	-6	5	1	1	1	0	0	0	1	-1	-1	0	1	-1	-1	0	1	-1	-1	0	1	-1	-1
OR	*nci*	2	1	1	-5	3	1	1	-6	5	1	1	1	0	0	0	1	-1	-1	0	1	-1	-1	0	1	-1	-1	0	1	-1	-1
	sl	*l*	s	s	*-l*	*l*	s	s	*-l*	*l*	s	s	s	0	0	0	s	-s	-s	0	s	-s	-s	0	s	-s	-s	0	s	-s	-s
OR	*sl*	*l*	s	s	*-l*	*l*	s	s	*-l*	*l*	s	s	s	0	0	0	s	-s	-s	0	s	-s	-s	0	s	-s	-s	0	s	-s	-s

Figure 9 The opening melody of Chopin's *Valse, Op. 18 (pci:* pitch-class interval, *nci:* name-class interval, *sl:* step-leap)

5. Conclusions

In this paper a number of general problems were presented regarding musical representation and pattern processing methodologies. A number of interesting integer pattern-matching problems were presented. Musical pattern 'significance' was also discussed and an attempt was made to formalise some interesting types of patterns for which pattern extraction algorithms can be developed. It is hoped that the problems discussed herein may contribute towards a better understanding of the distinctive qualities of musical pattern processing tasks and give rise to new useful and efficient pattern processing algorithms.

Acknowledgements

This research is part of the project Y99-INF, sponsored by the Austrian Federal Ministry of Science and Transport in the form of a START Research Prize.

References

Bregman, A. S. (1990) *Auditory Scene Analysis*. The MIT Press, Cambridge (Ma).

Cambouropoulos, E. (2000) From MIDI to Traditional Musical Notation. In *Proceedings of the AAAI Workshop on Artificial Intelligence and Music*, Austin, Texas (forthcoming).

Cambouropoulos, E., Crochemore, M., Iliopoulos, C.S., Mouchard, L. and Pinzon, Y.J. (1999) Algorithms for Computing Approximate Repetitions in Musical Sequences. In *Proceedings of the AWOCA'99 Workshop* (Australasian Workshop on Combinatorial Algorithms), Perth.

Cambouropoulos, E., Crawford, T. and Iliopoulos, C.S. (2000) Pattern Processing in Melodic Sequences: Challenges, Caveats and Prospects. *Computers and the Humanities*, 34:4 (forthcoming).

Cambouropoulos, E. (1998b) Musical Parallelism and Melodic Segmentation. In *Proceedings of the XII Colloquium of Musical Informatics*, Gorizia, Italy.

Crawford, T., Iliopoulos, C.S. and Raman, R. (1998) String Matching Techniques for Musical Similarity and Melodic Recognition. *Computing in Musicology*, 11:71-100.

Cope, D. (1990) Pattern-Matching as an Engine for the Computer Simulation of Musical Style. In *Proceedings of the International Computer Music Conference*, Glasgow.

Crochemore, M. (1981) An Optimal Algorithm for Computing the Repetitions in a Word. *Information Processing Letters*, 12(5):244-250.

Dovey, M.J. (1999) An Algorithm for Locating Polyphonic Phrases within a Polyphonic Musical Piece. In *Proceedings of the AISB99 Convention (Artificial Intelligence and Simulation of Behaviour)*, Edinburgh, U.K.

Iliopoulos, C.S., Moore, D.W.G. and Park, K. (1996) Covering a String. *Algorithmica*, 16:288-297.

McGettrick, P. (1997*) MIDIMatch: Musical Pattern Matching in Real Time*. MSc Dissertation, York University, U.K.

Reti, R. (1951) *The thematic Processes in Music*, The Macmillan Company, New York.

Rolland, P.Y., Ganascia, J.G. (1999) Musical Pattern Extraction and Similarity Assessment. In *Readings in Music and Artificial Intelligence*. E. Miranda. (ed.). Harwood Academic Publishers (forthcoming).