Pattern Induction and matching in polyphonic music and other multidimensional datasets

Dave Meredith
Department of Computing, City University, London
Northampton Square, London EC1V 0HB, UK

Geraint A. Wiggins
Department of Computing, City University, London
Northampton Square, London EC1V 0HB, UK

Kjell Lemström
Department of Computer Science, University of Helsinki
P.O.Box 26 (Teollisuuskatu 23), FIN-00014 University of Helsinki, FINLAND

Abstract

We present a new algorithm, SIA, which discovers maximal repeated patterns in any set of points in Cartesian spaces of any dimensionality. The worst-case running time of SIA is $O(kn^2 \log_2 n)$ for a k-dimensional dataset of size n.

SIATEC is an extension of SIA that generates a set of *translational equivalence classes* (TECs). If the input represents a musical score then each TEC contains all the transposition-invariant occurrences of a single maximal repeated pattern in the score. In the worst case, SIATEC takes time $O(kn^3)$ to compute the TEC of every maximal pattern computed by SIA.

We have also experimented with a set of heuristics, MU, that takes as input a dataset representing a musical surface together with the set of TECs generated by SIATEC for this dataset. MU computes a value for each TEC that is intended to represent the "musical significance" of the TEC. It then presents the TECs ordered according to this value.

The combined system of MU and SIATEC (which we call MUSIATEC) has been used to analyse some large-scale polyphonic works with very encouraging results.

We have also generalised SIA to produce a new pattern-matching algorithm. This algorithm, called SIA(M)ESE, takes as input a query pattern and a dataset and outputs a set of matches for the query pattern in the dataset. SIA(M)ESE is capable of true polyphonic music pattern matching in $O(kmn \log_2 n)$ time when looking for a k-dimensional pattern of size m in a k-dimensional text of size n. This makes SIA(M)ESE more efficient than existing algorithms for this purpose.

The work presented here is subject to patent protection.

1 Introduction

Many music analysts [13, 15] and music psychologists [10] agree that one of the most important steps in achieving a satisfying interpretation of a musical surface is the identification of important instances of repetition.

Our goal is to build a computational model of expert music cognition and it seems clear that one of the most important components in such a model would be an algorithm that can identify the most significant instances of repetition in a musical surface.

However, this brings with it an issue of computational efficiency. The datasets over which such a model must operate are significantly large. Therefore, it is important that the methods used are computationally efficient as well as logically correct.

In this paper, we present a new algorithm, SIATEC, which is to form the basis of such a model. It computes all the occurrences of every maximal repeated structure in any body of data. In the present application, the body of data we study will represent a musical score. The worst-case running time of SIATEC is $O(n^3)$ where n is the size (in notes) of the score to be analysed. We describe SIATEC in §3.

The output of SIATEC is in the form of sets of musical structures which are repeated (viewing transposed occurences as repetitions). But there are many such structures which would not normally be recognised as musically significant. In §4, we outline how heuristics can be used to select the more useful of these sets.

SIA(M)ESE (see section 5 below) is a generalisation of SIA (the part of SIATEC that computes the maximal repeated structures) which can be used for pattern *matching*. SIA(M)ESE is significantly more flexible than other approaches to the same task. In particular, it is capable, without modification, of finding polyphonic query patterns in polyphonic sources. In this case, the worst-case running time of SIA(M)ESE is $O(nm \log_2 n)$ for a query pattern containing m notes and a source containing n notes (assuming $m < n$). This is very competitive with other approaches (see §2.2). Furthermore, the output format for this application of the algorithm is in a form which we believe is more useful than that obtained from other comparable pattern-matching algorithms.

Finally, in §6, we propose some further applications of the SIA family of algorithms.

2 Related Work

2.1 Related Work on Pattern Discovery

Lerdahl and Jackendoff [10, p. 52] state that

> the importance of parallelism [that is, approximate or literal repetition] 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.

However they are unable to state precisely the conditions that must be satisfied by two passages before they can legitimately be "construed as parallel". Nonetheless, they are able to say that two passages that are "identical" (i.e. related to each other by exact or exact transposed repetition) "certainly count as parallel".

Lerdahl and Jackendoff are not the only authors to emphasize the identification of repetition (or 'parallelism') in interpreting music. Ian Bent [2] defines music analysis as

> the resolution of a musical structure into relatively simpler constituent elements, and the investigation of the functions of those elements within that structure.

Bent points out that "the phrase 'musical analysis,' taken in a general sense, embraces a large number of diverse activities" that "represent fundamentally different views of the nature of music" [2, p. 1]. Nevertheless, in his view, the "central activity" of music analysis is *comparison* and "the central analytical act is thus the test for identity" [2, p. 5].

In his *General Computational Theory of Musical Structure* (GCTMS), Emilios Cambouropoulos [3] presents what amounts to a computational model of his own interpretation of the semiotic method of analysis of Ruwet, Nattiez and their colleagues. In GCTMS a musical surface is first segmented according to local grouping principles embodied in a program called the 'Local Boundary Detection Module' [3, pp. 64–81]. The surface is then analysed by a 'Sequential Pattern Induction Algorithm' that identifies repetitions in the surface. A 'Selection Function' is then used to assign a value to each instance of repetition found by the Sequential Pattern Induction Algorithm. This value increases with the length of the pattern and its frequency of occurrence and decreases with the degree to which instances of the pattern overlap. Finally, the results generated by the Sequential Pattern Induction Algorithm and the Local Boundary Detection Module are used to produce a segmentation of the musical surface. This segmentation is given as input to another program called *Unscramble* which compares the segments in the segmentation and categorises them, placing similar segments into the same category.

In the computer science field of *stringology* (see §2.2) a considerable amount of work has been done on the development of algorithms for retrieving information from music databases. However, relatively little work has been done on machine *discovery* of repeated patterns in musical data and that which has been done [7, 14, for example] has been limited to the discovery of repetitions in monophonic databases or databases of polyphonic music in which the parts have been extracted and each part is represented as a separate string of intervals. Representing a polyphonic piece in this way means that a repetition discovery algorithm will fail to identify any instances of repetition in which the repeated pattern contains notes from more than one voice or part.

2.2 Related Work on Pattern Matching

The current state of the art in music information retrieval (MIR) has mostly been cast as a combinatorial pattern matching problem [4, 8].

Because the conventional string matching methods were originally developed for text matching, and texts do not share many of the features that are intrinsic to music, the whole framework needs to be modified in order to obtain musically meaningful matching results [8, 12]. The most salient feature of music that is not pertinent to text but is certainly pertinent to our study, is that real music is (almost) always polyphonic. Unmodified string-matching techniques are therefore not appropriate to polyphonic music matching.

Some approaches have been designed with polyphonic music in mind right from the start, though, strictly, they deal with a homophonic approximation to polyphonic music. The usual approach is to represent music as strings of chords $D' = D'_1 \cdots D'_{n'}$ each of whose elements contains simultaneous musical events.

Dovey [5] also allows parametrized spacing between the consecutive elements in the dataset, but the datasets he considers may be polyphonic. Because the algorithm tries to find an occurrence of the given template recursively in every possible location with every allowed spacing within the dataset, its time complexity becomes impractical for very large datasets; the worst case takes $O(n'm(t+1)^{(m-1)})$, where m is the size of the template, and n' and t are the length of the dataset in chords and the length of the allowed spacing, respectively. Moreover, to find transposed occurrences, the algorithm has to be reiterated for each possible transposition. In Dovey's algorithm, the template to be searched for may itself be polyphonic.

Uitdenbogerd and Zobel [16] combined ideas from music psychology with the straightforward application of conventional string matching methods. Their aim was to develop a heuristic capable of capturing the (monophonic) musical line that best represents a passage of polyphonic music. In their experiments with human listeners, a heuristic that always chooses the highest note of a chord performed the best. Thus, combining any conventional string matching method with their heuristic may be applied to the problem at the cost of losing information.

Holub *et al.* [6] presented an algorithm based on the so-called *bit-parallel* algorithmic technique. They started by using the well-known shift-or algorithm of Baeza-Yates and Gonnet [1] to find *multi-templates* (several templates combined in a single query) within monophonic datasets, and arrived at an algorithm capable of finding multi-templates within polyphonic datasets in $O(n'q)$ time[1] with a preprocessing phase taking $O(rm + |\Sigma'|)$ time. Here q, r and $|\Sigma'|$ denote the maximum number of events within any chord, the number of templates, and the number of symbols used in the templates, respectively. Holub *et al.* did not, however, consider transposition invariance. From a similar starting point, Lemström and Tarhio [9] introduced an algorithm, called MONOPOLY, capable of finding all transposed occurrences of a monophonic template within polyphonic datasets. The essential part of their algorithm runs in linear, $O(n')$ time[2] (with an $O(n'q)$ preprocessing and an $O(n'q(q+m))$ postprocessing phase).

3 Pattern Discovery: SIATEC

First, we present a new algorithm, SIATEC (structure induction algorithm with translational equivalence classes), which can discover complete sets of translation-invariant patterns in any set of points in an n-dimensional Cartesian space.

SIATEC takes such a multidimensional dataset as input and generates a set of *translational equivalence classes* (TECs). We say that two patterns in a dataset are *translationally equivalent* iff one can be obtained from the other by translation alone. Each pattern in a dataset is a member of exactly one TEC. The TEC to which a pattern belongs contains all and only those other patterns to which it is translationally equivalent. SIATEC generates for each of the largest repeated patterns in a dataset the TEC that contains that pattern.

The set of TECs generated by SIATEC for a musical surface represented as a multidimensional dataset typically contains the most significant repeated patterns. However, it also usually contains many instances of repetition that would not be considered musically significant by an expert music analyst.

In the rest of this section, we present the logic of SIA, the first stage of SIATEC, and an outline of the algorithm of the whole SIATEC process. Full details of the algorithms and their applications are given by Meredith *et al.* [11].

3.1 The Logic of SIA

SIA can be characterised as the computation of the structure set $\mathcal{S}$ given by equations (1) and (2), from the dataset, D. D is any set of points with any number of dimensions; the dimensions may be measured on any finitely expressible metric so long as it is possible to give a total ordering, $\prec$, on all the points in the vector space defined by D.

$$P = \{\langle p_1, V\rangle \mid p_1 \in D \wedge V = \{\overline{v} \mid \exists p_2 . p_2 \in D \wedge p_1 \prec p_2 \wedge \overline{v} = p_2 - p_1\}\} \quad (1)$$

$$\mathcal{S} = \{\langle \overline{v}_1, M\rangle \mid \exists p_1, V_1 . \langle p_1, V_1\rangle \in P \wedge \overline{v}_1 \in V_1 \wedge M = \{p_2 \mid \exists \overline{v}_2, V_2 . \langle p_2, V_2\rangle \in P \wedge \overline{v}_2 \in V_2 \wedge \overline{v}_1 = \overline{v}_2\}\} \quad (2)$$

The idea is to find all the maximal subsets of D, which are translated by a non-zero vector in the space defined by D's dimensions, to another subset of D. The ordering, $\prec$, prevents wastage of effort and duplication of results by removing repetition under symmetry. The output, S, is in the form of a set of $\langle$vector,point-set$\rangle$ pairs, relating each subset to the vector which translates it.

Note that there may be any finite number of intervening elements between the consecutive elements of a repeated pattern. In other words, SIA admits unlimited gaps in its patterns with no extra computing cost.

$\mathcal{S}$ is the input to the next stage of SIATEC, which computes the translational equivalence classes.

3.2 SIATEC

SIA alone simply computes every maximal repeated pattern in a dataset. SIATEC takes the output of SIA as input and then computes *all* the occurrences of each of the maximal repeated patterns computed by SIA. The operation of SIA and SIATEC is described and analysed in detail in Meredith et al.

[1] More precisely, the algorithm have a factor $\lceil \frac{m}{w} \rceil$ in its time complexity, where w denotes the size of computer words in bits. Thus, the best time complexity is achieved only in cases where $m \le w$.

[2] See Footnote 1.

3.3 Discovering Approximate Repetitions

So far, we have discussed only the discovery of exact repetitions. It is, of course, often desirable to discover (or match) inexact repetitions. There are two broad cases of approximate matching:

1. where the approximation can be accounted for in the notion of identity: for example, by considering modal transpositions instead of chromatic ones;

2. where the approximation is more radical, and cannot be considered as above.

In this second case, we argue there will often be *some* degree of identity, which, one might say, forms a *skeleton* of identity in which the approximation is framed. We speculate, therefore, that in this second, harder, case of approximate matching, we may be able to apply heuristic methods not dissimilar to those introduced in §5.3, below.

4 Musical Applications

4.1 Applying SIATEC to music

In order to apply SIATEC to musical scores, we represent the data on two dimensions, plotting pitch against onset time. Because the SIA family of algorithms is multidimensional, we can also add in other dimensions to represent, for example, duration, timbre and dynamics.

The choice of representation is significant. For example, representing pitch chromatically results in TECs containing structures identical under chromatic transposition, whereas a diatonic pitch representation allows SIATEC to find structures identical under diatonic transposition. Because SIATEC works on multidimensional files, it is possible to perform both of these comparisons (and, indeed, others as well) at the same time.

SIATEC can be feasibly applied to relatively large data files, since it runs in polynomial time, so we have applied it to files representing entire musical scores. While it is successful in finding many musically significant structures, it also finds many structures which would not usually be considered musically significant. For example, applying SIATEC to just the first two bars of the *Prelude in C Minor* (BWV871) of J. S. Bach's *Das Wohltemperirte Klavier* yields 133 TECs [11], most of which are musically insignificant. To address this issue, we are experimenting with heuristics to select muscially signficant TECs.

4.2 Heuristics for musical significance

We have developed a second program called MU that takes as input a dataset representing a musical surface together with the set of TECs generated by SIATEC for this dataset. MU uses a set of heuristics to calculate a value of "musical significance" for each TEC in the set generated by SIATEC for the dataset, and then sorts the TECs into of "musical significance" for output. Ultimately, we expect to develop different sets of heuristics for modelling different behaviours and applications.

Our first heuristic supposition is that, other things being equal, the signficance of a TEC increases with the size and frequency of repetition of the pattern. The second supposition is that patterns which significantly overlap are less likely to be perceived as individual structures. Finally, as an artifact of memory, a pattern is more likely to be significant if its elements are close together in time. We are currently experimenting with weightings of these aspects, and we have been successful in selecting significant structures in some datasets. However, more work is required before firm conclusions can be drawn.

5 Pattern Matching: SIA(M)ESE

We have also developed a new pattern-matching algorithm based on SIA. This algorithm, called SIA(M)ESE, takes as input a query pattern and a dataset and outputs a set of matches for the query pattern in the dataset; the query pattern and the dataset are both multidimensional sets of points, which, in the current study, are used to represent musical notes. In this section, we first state the logic of the algorithm and then outline how it can be efficiently implemented. We give some simple examples, and suggest that the output of SIA(M)ESE is more useful for pattern matching than that of similar approaches.

SIA(M)ESE is short for SIA (Matching) Express with Selection and Evaluation, and these aspects are outlined in this section. Full details of the algorithm and its application are described by Wiggins *et al.* [17].

5.1 The Logic of SIA(M)

SIA(M) differs only slightly from SIA as described in §3.1. In SIA(M), instead of computing the discovery set we compute the set $\mathcal{S}$, of repeated structures, we compute a match set, of representing occurences of a template, T in a dataset D using (3) and (4).

$$P = \{\langle p_1, V\rangle \mid p_1 \in T \wedge V = \{\overline{v} \mid \exists p_2 . p_2 \in D \wedge \overline{v} = p_2 - p_1\}\} \quad (3)$$

$$\mathcal{M} = \{\langle \overline{v}_1, M\rangle \mid \exists p_1, V_1 . \langle p_1, V_1\rangle \in P \wedge \overline{v}_1 \in V_1 \wedge M = \{p_2 \mid \exists \overline{v}_2, V_2 . \langle p_2, V_2\rangle \in P \wedge \overline{v}_2 \in V_2 \wedge \overline{v}_1 \simeq \overline{v}_2\}\} \quad (4)$$

There are just two small differences between these definitions and those given for SIA in §3. The first, and the more significant, is that in SIA(M), the vectors, $\overline{v}$, in P are not specified in terms of two points, p_1 and p_2, in the dataset, D, ordered under $\prec$ to avoid duplication under symmetry as shown in (1). Rather, the vectors are generated from two separate sets of points inhabiting the same vector space: the template, T, and the dataset, D. The separation of these two sets means that the efficiency measure of removing equivalents under symmetry in generating P

no longer makes sense, so the $\prec$ term of (1) is not present in (3).

The only difference between the subsequent calculations of $\mathcal{S}$ (2) and $\mathcal{M}$ (4) is that the final $=$ has been replaced by $\simeq$. This is to admit alternative notions of equality. For example, in the case where we are matching a human performance against an idealised, quantised database, we may want to relax equality a little to avoid having to deal with jitter explicitly in the match-finding process (see Section 5.3).

5.2 SIA(M)E

We have implemented this logic in an efficient program called SIA(M) Express. The worst-case running time of this program is $O(mn \log_2 n)$ but we have developed a special implementation that has an average running time of $O(mn)$. The algorithm computes all and only the non-strict subsets of the template, T, to be found within the dataset D paired with their locations in D.

The output of SIA(M)E returns strict matches without further processing, but partial matches require a little more work. The output format of SIA(M)E contains all *partial* matches in the sense that it locates all subsets of the template which have matched the dataset. The task is then to assemble these fragments into a good match. What is "good", of course, is domain dependent.

5.3 SIA(M)ESE

We have developed some simple heuristics, outlined in [17], which will assist in this task; there is insufficient space to present them here. It is likely that SIA(M) will require a different set of heuristics for each domain to which it is applied.

The class of matches generated by SIA(M)ESE for a given query pattern turns out to be highly appropriate for use in assessing student feedback in computer-based music and language education systems. We are currently assessing its use in such a system [18].

Because they allow arbitrary gaps in their matches, the SIA(M) family of algorithms is capable of finding templates at arbitrary levels of detail in the dataset – the level of detail found is simply that given in the template.

6 Further Work

Because of the generality of SIA, SIATEC and SIA(M)ESE, they are useful for a wide range of applications. Future work on application of the algorithms will include

- cognitive modelling of music perception;
- music transcription from audio signals (by application to Fourier spectra or acousmagraphs);
- music and sound analysis;
- data compression (including video and audio);
- video processing and image matching;
- music database indexing;
- protein structure analysis.

All of these potential applications are covered by the current patent on the algorithms presented here.

Acknowledgements

The work reported in this paper has been supported by EPSRC research grants GR/N08049 (Dave Meredith) and GR/R25316 (Kjell Lemström) and Finnish Academy grant #48313 (Kjell Lemström).

References

[1] R. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *Communications of the ACM*, 35(10):74–82, 1992.

[2] Ian Bent and William Drabkin. *Analysis*. New Grove Handbooks in Music. Macmillan, 1987.

[3] E. Cambouropoulos. *Towards a General Computational Theory of Musical Structure*. PhD thesis, Faculty of Music and Department of Artificial Intelligence, University of Edinburgh, 1998.

[4] T. Crawford, C. S. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11:71–100, 1998.

[5] M. J. Dovey. An algorithm for locating polyphonic phrases within a polyphonic musical piece. In *Proceedings of the AISB'99 Symposium on Musical Creativity*, pages 48–53, Edinburgh, 1999.

[6] J. Holub, C. S. Iliopoulos, B. Melichar, and L. Mouchard. Distributed string matching using finite automata. In *Proceedings of the 10th Australasian Workshop On Combinatorial Algorithms*, pages 114–128, Perth, 1999.

[7] Jia-Lien Hsu, Chih-Chin Liu, and Arbee L.P. 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.

[8] K. Lemström. *String Matching Techniques for Music Retrieval*. PhD thesis, Faculty of Science, University of Helsinki, Department of Computer Science, 2000. Report A-2000-4.

[9] K. Lemström and J. Tarhio. Detecting monophonic patterns within polyphonic sources. In *Content-Based Multimedia Information Access Conference Proceedings (RIAO'2000)*, pages 1261–1279, Paris, 2000.

[10] F. Lerdahl and R.S. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, MA., 1983.

[11] D. Meredith, K. Lemström, and G.A. Wiggins. Siatec and sia : Efficient algorithms for translation-invariant pattern discovery in multi-dimensional datasets, prep.

[12] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[13] J.-J. Nattiez. *Fondements d'une sémiologie de la musique*. Union Générale d'Editions, Paris, 1975.

[14] Pierre-Yves Rolland. Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4):334–350, 1999.

[15] N. Ruwet. *Langage, musique, poésie*. Editions du Seuil, Paris, 1972.

[16] A. L. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In *ACM Multimedia 98 Proceedings*, pages 235–240, Bristol, 1998.

[17] G.A. Wiggins, K. Lemström, and D. Meredith. SIA(M)ESE: An efficient algorithm for pattern matching in multidimensional datasets, prep.

[18] G.A. Wiggins and S. Trewin. A system for the concerned teaching of musical aural skills. In *Proceedings of ITS2000*, number 1839 in LNCS. Springer-Verlag, 2000.