Recherche de motifs pour l'étude critique de partitions musicales

Riyadh Benammar*, Christine Largeron**
Véronique Eglin*, Mylène Pardoen***

*Université de Lyon
CNRS INSA-Lyon, LIRIS, UMR5205, F-69621, France
riyadh.benammar@insa-lyon.fr, veronique.eglin@insa-lyon.fr,
**UJM-Saint-Etienne, CNRS,
Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France
christine.largeron@univ-st-etienne.fr
***Université Lumière Lyon 2,
Maison des sciences de l'homme Lyon Saint-Etienne (MSH - LSE)
mylene.pardoen@wanadoo.fr

Abstract. Music score analysis is an ongoing issue for musicologists. Discovering frequent musical motifs with variants is needed in order to make critical study of music scores and investigate compositions styles. We introduce a mining algorithm, called CSMA for **C**onstrained **S**tring **M**ining **A**lgorithm), to meet this need considering symbol-based representation of music scores. This algorithm, through motif length and maximal gap constraints, is able to find identical motifs present in a single string or a set of strings. It is embedded into a complete data mining process aiming at finding variants of musical motif. Experiments, carried out on several datasets, showed that CSMA is efficient as string mining algorithm applied on one string or a set of strings.

1 Introduction

Musical motifs are pieces of music that can define a signature for a composer, a music score or a music style. They correspond to identical repeating music chunks or variations applied on a part of music. As music notes are characterized by three kinds of information (melodic, rhythmic and harmonic), musical motifs can also be melodic and/or rhythmic and/or harmonic and our goal is to extract such motifs, with variants, from music scores transcriptions. In data mining, this task corresponds to motif mining from a single sequence or a set of sequences called strings. In our framework, we are more interested in motif mining on strings than by pattern mining in sequences since we consider that a music score can be represented by one or several sequences, one sequence per instrument, and at each timestamp there is only one note in the sequence; the harmonic information is not considered. Moreover, a musical motif corresponds to a music chunk appearing at minimal number of positions through the music score. This motif can be melodic, rhythmic or both, depending on the nature of the music event features.

Among the first works in the music domain, Hsu *et al.* introduced a method to identify frequent motifs in a music score, by considering only the melodic information in a single sequence Hsu et al. (1998). Liu *et al.*Liu et al. (1999) proposed an improvement of this method aiming at finding all non-trivial motifs (*i.e.* motifs that do not have sub motifs with the same frequency). These algorithms are not suited for our task since we are interested in identifying motifs with gaps whereas they consider only contiguous motifs. Moreover they only process one sequence while we consider music scores with one or several instruments corresponding to a set of sequences, one per instrument. However, we retain from the algorithm proposed by Liu *et al.* the structure for coding the motifs, that we adapt to handle gaps.

Besides, other works study the representation of the music scores. When music data is in audio format, pitch values can be firstly extracted. Then, melodic motifs can be identified using for instance, episode mining approach like in Liu et al. (1999). To exploit both melodic and rhythmic information, Béatrice Fuchs suggests to transform the input music data into a data stream for mining frequent itemsets Fuchs (2012). Finally, the work the most related to ours has been done by Jiménez *et al.* who designed an algorithm able to find transposed musical motifs by exact matching Jiménez et al. (2011) but, in our work, we consider three variants of an initial motif: transposed, inverted and mirror forms. Indeed, all these forms can define a signature for the composer, the score or the music style that can be used as features for other mining tasks such as supervised or non supervised clustering.

We present in the next section a new algorithm, **CSMA**, able to extract motifs from one or several strings with constraints related to the minimal frequency, gaps inside motifs, and motif length. Section 3 is dedicated to the preprocessing, that we propose, for extracting three variants of an initial motif. Section 4 presents experiments carried out on a real dataset to illustrate the interest of our approach and, Section 5 concludes the paper.

2 Constrained String Mining Algorithm

CSMA (**C**onstrained **S**tring **M**ining **A**lgorithm) has been designed for discovering all frequent motifs in a string Benammar et al. (2017). CSMA performs motifs search according to constraints related to frequency, gaps between motifs, minimal and maximal length of motifs. A motif m_i is defined by three elements $m_i = (X, freq(X), P_i = [(p_{i1}, len_{i1}), ..., (p_{in}, len_{in})])$ such that X corresponds to the motif value (ordered list of items), $freq(X)$ corresponds to its frequency and P_i its positions and lengths. In the set of positions, called P_i, the j^{th} position of the i^{th} motif is denoted p_{ij} and its length at this position is denoted len_{ij}.

The pseudo-code of **CSMA** is given in Algorithm 1. This algorithm takes in input a sequence S, a minimum frequency threshold $minFrequency$, a maximum allowed gap length inside motifs $maxGap$, a minimum motif length $minLength$ and a maximum motif length $maxLength$.

The first step of CSMA (Line 4, Algorithm 1) consists in computing the set $\mathcal{F}_1$ containing the frequent motifs of length one using **COMPUTE** function. The items with frequency greater than or equal to $minFrequency$ are added to $\mathcal{F}_1$. In order to get the set $\mathcal{F}_K$ containing the motifs of length equal to ($K = 2$), a joining operation **JOIN** is considered (line 7) between each element m_i of $\mathcal{F}_{K-1}$ and each item m_j belonging to $\mathcal{F}_1$. The joining operation is $\mathcal{O}(|P_i| \times |P_j|)$. So, in order to prune the search space, we compute the position on which the motif m_i is considered as frequent (line 8). This position, called $frequentPosition$, corre-

Algorithm 1: Constrained String Mining Algorithm (CSMA)

```
Input  : Sequence S, minFrequency,maxGap, minLength and maxLength
Output: $\mathcal{F}$: The set of frequent motifs respecting constraints
1  begin
2      $K = 1$;
3      $\mathcal{F}_1 = \emptyset$;
4      COMPUTE($\mathcal{F}_1$);
5      while $\mathcal{F}_K \neq \emptyset$ do
6          $K = K + 1$;
7          for $m_i = (X, freq(X), P_i) \in \mathcal{F}_{K-1}$ do
8              $frequentPosition = p_{iminFrequency} + len_{iminFrequency}$;
9              $\mathcal{C}$ = GEN_CANDIDATES($frequentPosition, \mathcal{F}_1$);
10             for $m_j = (Y, freq(Y), P_j) \in \mathcal{C}$ do
11                 $m_l$ = JOIN($m_i, m_j, maxGap, maxLength$); if
                    $freq(Z) \geq minFrequency$ then
12                     $\mathcal{F}_K = \mathcal{F}_K \cup \{m_l\}$;
13                 end
14             end
15         end
16     end
17     $\mathcal{F} = \bigcup_{k \leq K} \mathcal{F}_k$
18     FILTER($\mathcal{F}, minLength$);
19     return $\mathcal{F}$;
20 end
```

sponds to the sum of the index of $m_i \in \mathcal{F}_{K-1}$ at the $minFrequency^{th}$ position and the length of m_i for the same position. Then, candidate motifs are generated using the **GEN_CANDIDATES** function. The interested reader is referred to the original article for a detailed description of this function Benammar et al. (2017). Once the selection of candidate motifs is done, the joining operation is performed for the selected motif m_i with each element $m_j \in \mathcal{C}$. The motif joining (concatenation) is defined as follows:
Let be two motifs $m_1 \in \mathcal{F}_{K-1}$ and $m_2 \in \mathcal{F}_1$ defined as $m_1 = (X, freq(X),\ P_1 = [\bigcup_{i \leq freq(X)} (p_{1i}, len_{1i})])$ and $m_2 = (Y, freq(Y), P_2 = [\bigcup_{j \leq freq(Y)} (p_{2j}, len_{2j})])$, m_1 join m_2 gives $m_3 \in \mathcal{F}_K$ defined as $m_3 = (Z, freq(Z), P_3)$ such that Z is the concatenation of (X, Y) and P_3 is a set of positions p_{3k} and lengths len_{3k}. A position $p_{3k} \in P_3$ equal to p_{1i} if and only if $\exists j \leq freq(Y)$ such that the three conditions are verified:

$$\begin{cases} 0 \leq p_{2j} - (p_{1i} + len_{1i}) \leq maxGap & (1) \\ i = \arg\min_{l \leq freq(X)} (p_{2j} - (p_{1l} + len_{1l})) & (2) \\ p_{2j} + len_{2j} - p_{1i} \leq maxLength & (3) \end{cases}$$

The positions p_{1i} from m_1 and p_{2j} from m_2 verifying the three conditions allow to define the position p_{3k} corresponding to p_{1i} for m_3, and the length len_{3k} is equal to $p_{2j} + len_{2j} - p_{1i}$. The frequency of m_3 is equal to the number of positions in P_3. It can be noticed that the frequency of each new motif is lower or equal to its sub-motifs. This means that the joining operation verifies the anti-monotony property which allows to prune the search space. Once $\mathcal{F}_2$ is obtained, the other sets $\mathcal{F}_K$ of length $K > 2$, are computed and the while loop stops when no new motif is generated. In the next step, (line 19 in Algorithm1), all frequent motifs of order $k \leq K$ are put in $\mathcal{F}$. Then, motifs that do not respect the *minLength* constraint are removed from $\mathcal{F}$. The **FILTER** function scans each motif $m \in \mathcal{F}$ and if it finds a position for which len_i is lower than *minLength* it removes it from the set P. In the end, the value $freq(X)$ is updated and if it is lower than *minFrequency* the motif is removed from $\mathcal{F}$. As Algorithm 1 makes a breadth first search to build motifs, it needs an exponential running time which is estimated to $O((max(|P|) \times |F_1)|^{maxLength})$; with $max(|P|)$ the maximal size of positions sets.

The previous algorithm searches motifs in a single string. However, for a given piece of music, we can be interested in identifying motifs for different instruments namely in several sequences, one per instrument. One can notice, that CSMA can be extended, in an easy way, to extract motifs within a string database. The adaptation is done in the joining operation by adding to the first conditions a new one according to which "p_{1i} and p_{2j} should belong to the same string". In the sequel, to make difference between the two versions, we call the first one CSMA1 and the adapted one CSMA2.

3 Music motifs variants extraction

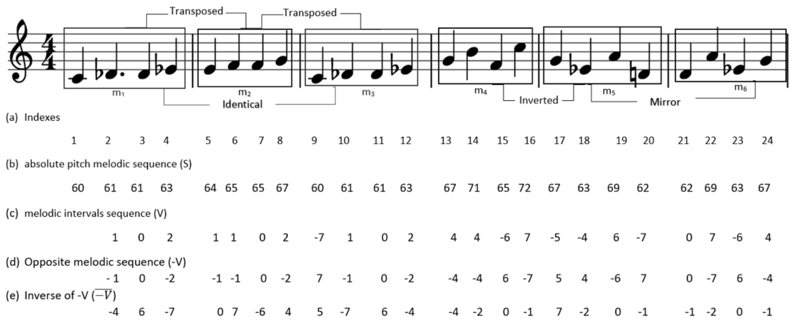
The aim of our work is to identify musical motifs as well as three musical variants of a motif: its transposed, inverted or mirror form as illustrated on Fig. 1. These forms are useful to characterize a composer or a music score. We describe below a preprocessing of a melodic sequence S in MIDI format, allowing CSMA to extract these variants.

We consider the sequence of intervals between consecutive notes from S. By this way, three new sequences are built as illustrated on Fig. 1. The first one, denoted V corresponds to the melodic variation between two notes. The second sequence, denoted $-V$ is obtained by taking the opposite of each value in the sequence V. Finally, the last sequence, called inverse of $-V$ and denoted $\overline{-V}$ is obtained by taking the sequence $-V$ in reverse order beginning by its last element (cf. Fig. 1). Once the primary and secondary sequences have been defined, CSMA can be applied to them to extract musical motifs variants, as explained below.

To detect transposed and inverted motifs, $-V$ is put after V (i.e. the last element of V is followed by the first element of $-V$) this makes a sequence $< V, -V >$ of size $2l_v$ on which CSMA is applied. Then, for each motif with two positions $(i, len_i), (j, len_j)$ such that *if* $(i \leq l_v) \wedge (j \leq l_v) \wedge (len_i = len_j) \wedge$ *(there is no identical motif occurrence on these positions)* then we have a *transposed form* at positions i and j.

If CSMA extracts a motif having two positions i and j, if $i \leq l_v, j > l_v$ and $S_{i-1} = S_{j-l_v}$ then the subsequence $S_1 =< S_{j-l_v}, ..., S_{j-l_v+len_j} >$ is an *inverted form* of the subsequence $S_2 =< S_{i-1}, ..., S_{i-1+len_i} >$.

To detect mirror form, $\overline{-V}$ is put after V such that the last element of V is followed by the first element of $\overline{-V}$. This makes again a new sequence $< V, \overline{-V} >$ of size $2l_v$. If CSMA finds a motif $m = (X, freq(X), P)$ with $(i, len_i), (j, len_j) \in P$ such that $(i \leq l_v) \wedge (j > l_v) \wedge (V_{i+len_i/2} = 0) \wedge (len_i = len_j)$ then the subsequence from position $i-1$ to $i+len_i-1$ in the melodic sequence S is a *mirror motif.*



(a) Indexes

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

(b) absolute pitch melodic sequence (S)

60 61 61 63 64 65 65 67 60 61 61 63 67 71 65 72 67 63 69 62 62 69 63 67

(c) melodic intervals sequence (V)

1 0 2 1 1 0 2 -7 1 0 2 4 4 -6 7 -5 -4 6 -7 0 7 -6 4

(d) Opposite melodic sequence (-V)

-1 0 -2 -1 -1 0 -2 7 -1 0 -2 -4 -4 6 -7 5 4 -6 7 0 -7 6 -4

(e) Inverse of -V ($\overline{-V}$)

-4 6 -7 0 7 -6 4 5 -7 6 -4 -4 -2 0 -1 7 -2 0 -1 -1 -2 0 -1

FIG. 1: musical motifs and their variants

Music score	Part	Melodic sequence		Melodic variations			Rhythmic sequence		Pitch sequence
		size	simple motifs	transposed motifs	inverted motifs	mirror motifs	size	simple motifs	simple motifs
score 1	P1	287	13	5	3	3	307	58	39
	P2	295	17	17	1	0	313	56	61
	P3	239	19	2	2	4	270	66	5
	P4	217	41	5	1	14	240	56	143
score 2	P1	66	56	55	0	2	141	34	68
	P2	166	7	11	0	1	244	77	28
	P3	518	55	24	6	9	586	138	154
	P4	129	16	21	0	4	182	55	47
	P5	458	73	21	0	1	475	98	90

TAB. 1: Number of distinct musical motifs (and variants) in real music scores

4 Evaluation of CSMA

The comparison of CSMA with Liu algorithm Liu et al. (1999) on large artificial datasets has confirmed its efficiency Benammar et al. (2017). In this paper, we report an evaluation done on real music scores.

CSMA has been tested on two music scores 'The art of fugue Bach BWV 1080' (score 1) and 'Johann Pachelbel hexachordum apollinis' (score 2) in midi format. Firstly, each music score has been transformed into a set of symbolic sequences: a sequence per instrument. Thus we obtained four sequences, P1 to P4, for score 1 and five for score 2 (P1 to P5). Then, different types of sequences have been extracted: absolute pitch sequence (MIDI value-based melodic sequence), duration sequence (rhythmic sequence) and pitch sequence (melodic and rhythmic sequence). Melodic sequences allow to extract simple motifs with or without variations (transposed, inverted and mirror forms) whereas only simple motifs can be detected in the other sequences.

The parameters *minFrequency*, *maxLength*, *maxGap* and *minLength* were respectively fixed to 2, maximum Java integer value (no constraint for *maxLength*), 0 and 4 for simple motifs and 3 for variants. The number of motifs extracted for each sequence is given in Table 1.

We can notice that both music scores contain all types of motifs even if parts in score 1 are of the same size whereas score 2 contains mix long and short parts. However, melodic variants appear across the different parts in score 1. We can conclude that there is a general theme hidden through the parts in score 1. The distribution of the motifs in score 2 is different. Score

2 uses more transposed motifs, notably in part 1 (P1) even if this part is the shortest. Part 3 contains the different forms. That is not the case of part 5 even if they have approximatively the same size. This difference in the motif distribution confirms our hypothesis concerning the discriminant power of these forms used as descriptive features of music scores for composers work identification.

5 Conclusion and Future Work

In this paper, we introduced an original motif mining algorithm, called CSMA, able to find contiguous and non-contiguous motifs. This algorithm incorporates constraints related to frequency, gap size and motif length. With its two versions, CSMA can find motifs from one or multiple strings. One of its advantage is that it saves the motif positions in the string and offers the possibility to find motifs with gaps. Those positions are, then, useful to extract musical motifs variants such as transposed, inverted and mirror forms. It should be pointed out that these positions are also useful for the expert in his analysis of the music scores.

References

Benammar, R., C. Largeron, V. Eglin, and M. Pardoen (2017). Discovering motifs with variants in music databases. In *International Symposium on Intelligent Data Analysis*, pp. 14–26. Springer.

Fuchs, B. (2012). Co-construction interactive de connaissances, application à l'analyse mélodique. In *IC 2011, 22èmes Journées francophones d'Ingénierie des Connaissances*, pp. 705–722.

Hsu, J.-L., A. L. Chen, and C.-C. Liu (1998). Efficient repeating pattern finding in music databases. In *Proceedings of the seventh ICIKM, ACM*, pp. 281–288.

Jiménez, A., M. Molina-Solana, F. Berzal, and W. Fajardo (2011). Mining transposed motifs in music. *Journal of Intelligent Information Systems 36*(1), 99–115.

Liu, C.-C., J.-L. Hsu, and A. L. Chen (1999). Efficient theme and non-trivial repeating pattern discovering in music databases. In *Proceedings, 15th International Conference on Data Engineering, IEEE*, pp. 14–21.

Résumé

L'analyse des partitions musicales est un problème récurrent pour les musicologues, en particulier, la découverte de motifs musicaux avec des variantes est nécessaire pour faire une étude critique des partitions et étudier les styles de composition. Dans ce but, nous proposons un algorithme de fouille de données appelé CSMA, pour Constrained String Mining Algorithm. Cet algorithme est capable de trouver des motifs identiques présents dans une seule séquence ou dans une base de séquences. Intégré à un processus complet d'extraction de connaissances à partir de données, il permet aussi de trouver des variantes des motifs musicaux. Les expériences réalisées sur plusieurs bases de données, ont montré l'efficacité de CSMA.