

# Discovering Motifs with Variants in Music Databases

Riyadh Benammar<sup>1,2(✉)</sup>, Christine Largeron<sup>1,3</sup>, Véronique Eglin<sup>1,2</sup>,  
and Myléne Pardoën<sup>4</sup>

<sup>1</sup> Université De Lyon, Lyon, France

<sup>2</sup> CNRS INSA-Lyon, LIRIS, UMR5205, 69621 Lyon, France

`{riyadh.benammar, veronique.eglin}@insa-lyon.fr`

<sup>3</sup> UJM-Saint-Etienne, CNRS, Institut d'Optique Graduate School,  
Laboratoire Hubert Curien UMR 5516, 42023 Saint-Etienne, France

`christine.largeron@univ-st-etienne.fr`

<sup>4</sup> Institut des Sciences de l'Homme (FRE 3768), 14 Avenue Berthelot,  
69363 Lyon cedex 07, France

`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.

**Keywords:** Music scores analysis · Music motif mining · String mining

## 1 Introduction

Everyone is able to tell sometimes: ‘This song seems to be from Supertramp or this music from Chopin’. By listening to a musical excerpt one can recognize the singer or the music’s style, even if the singer is unknown. For an instrumental piece we are able to recognize the composer. This is probably due to the motifs appearing in the music score. 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 that is modelled by a string. 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. To the best of our knowledge, only few works tried to extract musical motifs where the mining process is applied on one or many sequences of musical symbols. However, musical motif extraction

with variants, except transposed motifs, is a case-study that was not addressed before. In our work we are interested in melodic and/or rhythmic motifs mining with variants.

Our goal is to extract musical motifs 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 this paper, we propose an algorithm, called Constrained String Mining Algorithm (CSMA), able to solve this task. In addition, in order to find musical motif variants, we present a preprocessing of the music data. This preprocessing provides different representations of a music score offering to CSMA the possibility to identify three variants of an initial motif: transposed, inverted and mirror forms. Experiments carried out on synthetic and real datasets have confirmed the interest of the proposed approach.

The rest of the paper is organized as follows. Section 2 is dedicated to the state of the art, while the proposed algorithm CSMA is presented in Sect. 3. The experiments are described in Sect. 4 and Sect. 5 concludes the paper.

## 2 Related Work

Agrawal and Srikant introduced sequence mining methods, based on their well-known algorithm Apriori [1], initially designed for itemset mining in a transactional database where each transaction consists in a customer-id, the transaction time, and the items bought [2]. Formally, the set of items is defined by  $I = \{i_1, \dots, i_n\}$  and a sequence  $s$  over  $I$  is an ordered list  $\langle s_1 \dots s_l \rangle$ , where  $s_i \subset I$  ( $1 \leq i \leq l, l \in \mathbb{N}$ ) is an itemset.  $l = |s|$  is called the size of the sequence [3]. A pattern is a subsequence with multiple occurrences in the database. It is considered as frequent if it appears at least *minFrequency* time where *minFrequency* is a threshold fixed by the user.

Several algorithms were proposed to solve efficiently this task like FreeSpan and PrefixSpan, based on pattern-growth methods [4, 5] or SPADE which implements a vertical format-based mining approach [6]. Among the most recent sequence mining algorithms that outperform the other approaches, we can mention CM-SPADE, an extended version of SPADE that can incorporate constraints. This algorithm is based on co-occurrence MAP structure allowing a good pruning strategy during the candidate generation. Some other works introduced constraint-based sequential pattern mining approaches, like cSpade [7] and Prefix-Growth [8], where a set of constraints related to pattern length, gaps inside pattern and other parameters are set by the user.

In our framework, we are more interested in motif mining on strings than by pattern mining in sequences. Indeed, motif mining is applied on a string dataset, which is equivalent to a sequence dataset with itemsets of length one. This corresponds to our case where we consider that a music score can be represented by one or several sequences, one sequence per instrument, and where at each timestamp there is only one note in the sequence; the harmonic information is not considered.

In such sequences, three types of motifs can be extracted: *contiguous motifs*, *non-contiguous motifs* and *approximate contiguous motifs*. Contiguous motifs

are motifs where the elements are lined up behind each other whereas in non-contiguous motifs, jumps between elements positions are allowed. In approximate contiguous motifs, introduced by Floratou et al., the presence of a certain controlled noise is permitted but this notion is out of the scope of this work which focuses only on contiguous and non-contiguous motifs [9].

Finally, we can mention episode mining which exploits sequences of events where each event has an associated time of occurrence. To detect events occurring frequently close to each other, episode mining considers a sliding overlapping window of fixed size. An event is considered as frequent if it occurs in a minimal number of windows [10]. For more details, we refer the interested reader to [11].

In our framework, 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 [12]. Liu *et al.* [13] 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 [13]. To exploit both melodic and rhythmic information, Béatrice Fuchs suggests to transform the input music data into a data stream and then, mining frequent itemset [14]. 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 [15]. In this approach, the song is transformed into a sequence of notes and the motifs are extracted by a sequence mining algorithm, called SSMiner. In the same way than in this last work, we are interested in finding motifs as well as their variations. For a given motif we can have three possible variations: transposed, inverted and mirror forms. All these forms are interesting for the musicologist since they reflect the style of a composer. Consequently, they 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 propose 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. In **CSMA**, the motif positions in the music score are saved. Those positions are helpful for the musicologist to analyse the score. They are also exploited to build new representations of the music score which are used to extract variants of the musical motifs.

### 3 Contribution

In this section, we introduce a new algorithm, called **CSMA** (Constrained String Mining Algorithm), for discovering all frequent motifs in a string. CSMA performs motifs search according to constraints related to frequency, gaps between motifs, minimal and maximal length of motifs. It uses the same structure as the algorithm of Liu *et al.* for coding motifs but with some modifications to take gaps into account [13].

Hence, a motif  $m_i$  is defined by three elements  $m_i = (X, freq(X), P_i = [(p_{i1}, len_{i1}), \dots, (p_{in}, len_{in})])$  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  $len_{ij}$ .

*Example 1.* Given the sequence  $S = \langle ABABCD CABDCE \rangle$ :

- The motif  $m$  corresponding to  $A$  is defined by  $m=(A, freq(A)=3, P = [(1,1), (3,1), (8,1)])$ ;

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  $maxLength$ .

Table 1 shows an example of the process on the sequence of *Example 1* with  $minFrequency = 2$ ,  $maxGap = 1$ ,  $minLength = 1$  and  $maxLength = 4$ .

The first step of CSMA (Line 4, Algorithm 1) consists in computing the set  $\mathcal{F}_1$  containing the frequent motifs of length one. It is performed using the function **COMPUTE** which starts by enumerating all possible items from  $S$  and computing the frequency of each item. The items with frequency greater than or equals to  $minFrequency$  are added to  $\mathcal{F}_1$ .

In the example of Table 1, the set of items is  $I = \{A, B, C, D, E\}$  and  $\mathcal{F}_1 = \{m_1 = (A, 3, P_1=[(1,1), (3,1), (8,1)]), m_2 = (B, 3, P_2=[(2,1), (4,1), (9,1)]), m_3 = (C, 3, P_3=[(5,1), (7,1), (11,1)]), m_4 = (D, 2, P_4=[(6,1), (10,1)])\}$ ; the motif containing  $E$  does not belong to  $\mathcal{F}_1$  because it does not appear at least  $minFrequency$  times. In order to get the set  $\mathcal{F}_K$  containing the motifs of length equals 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*, corresponds 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.

*Example 2.* In Example 1, as  $minFrequency = 2$ , the  $minFrequency^{th}$  position will be the second position. Consequently, for the motif  $m_1 = (A, 3, [(1,1), (3,1), (8,1)])$  knowing that its second position corresponds to  $p_{12} = 3$  and its length at this position is  $len_{12} = 1$ , *frequentPosition* of  $m_1$  equals  $p_{12} + len_{12} = 3 + 1 = 4$ .

**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( $\mathcal{F}_1$ ); /* Compute frequent motifs of length 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        $\mathcal{C} = \mathbf{GEN\_CANDIDATES}(frequentPosition, \mathcal{F}_1)$ ;
10      for  $m_j = (Y, freq(Y), P_j) \in \mathcal{C}$  do
11         $m_l = \mathbf{JOIN}(m_i, m_j, maxGap, maxLength)$ ;
12        /*  $m_l = (Z, freq(Z), P_k)$  is a new motif built by joining  $m_i$  and  $m_j$  */
13        if  $freq(Z) \geq minFrequency$  then
14           $\mathcal{F}_K = \mathcal{F}_K \cup \{m_l\}$ ;
15        end
16      end
17    end
18     $\mathcal{F} = \bigcup_{k \leq K} \mathcal{F}_k$ 
19    FILTER( $\mathcal{F}, minLength$ ); /* This function removes motifs that violates  $minLength$  frequency constraints and keeps only frequent ones */
20    return  $\mathcal{F}$ ;
21 end

```

---

Then, candidate motifs are generated using the **GEN\_CANDIDATES** function. Given the position  $frequentPosition$  and the set of all frequent motifs of length one already extracted, this function computes a set  $\mathcal{C} \subseteq \mathcal{F}_1$  of candidate motifs that could be joined to  $m_i$  starting from  $frequentPosition$ . Our pruning strategy is based on the fact that a motif  $m_j \in \mathcal{F}_1$  cannot be a candidate for  $m_i \in \mathcal{F}_{K-1}$  if it does not appear after  $frequentPosition$  since the joining result of  $m_i$  and  $m_j$  cannot be frequent.

For instance, in example of Table 1, for the motif  $m_4(D)$ , as the motifs  $A$ ,  $B$  and  $D$  do not appear after its  $frequentPosition$ , which equals to 11,  $DA$ ,  $DB$  and  $DD$  could not be frequent and, the only candidate for this motif is  $C$ .

Once the selection of candidate motifs is done, a set  $\mathcal{C} \subseteq \mathcal{F}_1$  is computed. Then, 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  $len_{3k}$ .

**Table 1.** Processing of  $S = \langle ABABCD CABDCE \rangle$  with  $minFrequency = 2$ ,  $maxGap = 1$  and  $maxLength = 4$ 

$K$	$k$	$\mathcal{F}_{K-1}$	Frequent Position	$C$	New motif	Accepted	Violations
-	-	$\emptyset$	-	-	-		
1	-	$A, 3, [(1, 1), (3, 1), (8, 1)]$	-	-	-	✓	
1	-	$B, 3, [(2, 1), (4, 1), (9, 1)]$	-	-	-	✓	
1	-	$C, 3, [(5, 1), (7, 1), (11, 1)]$	-	-	-	✓	
1	-	$D, 2, [(6, 1), (10, 1)]$	-	-	-	✓	
2	1	$A, 3, [(1, 1), (3, 1), (8, 1)]$	4	$A$	$A * A, 1, [(1, 3)]$	X	$minFrequency$
2	1	$A, 3, [(1, 1), (3, 1), (8, 1)]$	4	$B$	$A * B, 3, [(1, 2), (3, 2), (8, 2)]$	✓	
2	1	$A, 3, [(1, 1), (3, 1), (8, 1)]$	4	$C$	$A * C, 1, [(3, 3)]$	X	$minFrequency$
2	1	$A, 3, [(1, 1), (3, 1), (8, 1)]$	4	$D$	$A * D, 1, [(8, 3)]$	X	$maxGap, minFrequency$
2	2	$B, 3, [(2, 1), (4, 1), (9, 1)]$	5	$A$	$B * A, 1, [(2, 2)]$	X	$maxGap, minFrequency$
2	2	$B, 3, [(2, 1), (4, 1), (9, 1)]$	5	$B$	$B * B, 1, [(2, 3)]$	X	$maxGap, minFrequency$
2	2	$B, 3, [(2, 1), (4, 1), (9, 1)]$	5	$C$	$B * C, 2, [(4, 2), (9, 3)]$	✓	
2	2	$B, 3, [(2, 1), (4, 1), (9, 1)]$	5	$D$	$B * D, 2, [(4, 3), (9, 2)]$	✓	
2	3	$C, 3, [(5, 1), (7, 1), (11, 1)]$	8	$A$	$C * A, 1, [(7, 2)]$	X	$minFrequency$
2	3	$C, 3, [(5, 1), (7, 1), (11, 1)]$	8	$B$	$C * B, 1, [(7, 3)]$	X	$minFrequency$
2	3	$C, 3, [(5, 1), (7, 1), (11, 1)]$	8	$C$	$C * C, 1, [(5, 3)]$	X	$maxGap, minFrequency$
2	3	$C, 3, [(5, 1), (7, 1), (11, 1)]$	8	$D$	$C * D, 1, [(5, 2)]$	X	$maxGap, minFrequency$
2	4	$D, 2, [(6, 1), (10, 1)]$	11	$C$	$D * C, 2, [(6, 2), (10, 2)]$	✓	
3	1	$A * B, 3, [(1, 2), (3, 2), (8, 2)]$	5	$A$	$A * B * A, 1, [(1, 3)]$	X	$maxGap, minFrequency$
3	1	$A * B, 3, [(1, 2), (3, 2), (8, 2)]$	5	$B$	$A * B * B, 1, [(1, 4)]$	X	$maxGap, minFrequency$
3	1	$A * B, 3, [(1, 2), (3, 2), (8, 2)]$	5	$C$	$A * B * C, 2, [(3, 3), (8, 4)]$	✓	
3	1	$A * B, 3, [(1, 2), (3, 2), (8, 2)]$	5	$D$	$A * B * D, 2, [(3, 4), (8, 3)]$	✓	
3	2	$B * C, 2, [(4, 2), (9, 3)]$	12	-	-		
3	2	$B * D, 2, [(4, 3), (9, 2)]$	11	$C$	$B * D * C, 2, [(4, 4), (9, 3)]$	✓	
3	3	$D * C, 2, [(6, 2), (10, 2)]$	12	-	-		
4	1	$A * B * C, 2, [(3, 3), (8, 4)]$	12	-	-		
4	2	$A * B * D, 2, [(3, 4), (8, 3)]$	11	$C$	$A * B * D * C, 2, [(3, 5), (8, 4)]$	X	$maxLength, minFrequency$
4	3	$B * D * C, 2, [(4, 4), (9, 3)]$	12	-	-		

A position  $p_{3k} \in P_3$  equals 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 first condition guarantees the constraint associated to *maxGap*. It allows to compute all possible gap values for a given  $p_{2j}$  considering all  $p_{1i}$ . Only positive gap values that are lower or equals to *maxGap* are retained. Then, with the second condition, the index  $i$  of  $p_{1i}$  which has a minimal value for the gap is recovered. Finally, in the third condition, the length of the motif is computed in order to check if it respects the *maxLength* constraint.

More precisely, the positions  $p_{1i}$  from  $m_1$  and  $p_{2j}$  from  $m_2$  verifying the previous 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 conclusion, in this example we obtain the following result:  $\mathcal{F}_1 = \{A, B, C, D\}$ ,  $\mathcal{F}_2 = \{A * B, B * C, B * D, D * C\}$ ,  $\mathcal{F}_3 = \{A * B * C, A * B * D, B * D * C\}$  and  $\mathcal{F}_4 = \emptyset$ , where  $*$  denotes a gap.

In the next step, (line 19 in Algorithm 1), 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  $m = (X, freq(X), P = [\bigcup_{i \leq freq(X)} (p_i, len_i)]) \in \mathcal{F}$  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  $\mathcal{F}$ . In the example, as *minLength* equals to 1,  $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$ . In Table 1, the column *violation* shows violated constraints that fails the joining operation.

As Algorithm 1 makes a breath 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.

### 3.1 Extension of CSMA

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 noticed, that CSMA can be extended, in a 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.

## 4 Evaluation of CSMA

Our experiments aims to evaluate the results provided by CSMA on synthetic and real datasets.

#### 4.1 Experiments on Synthetic Dataset

In order to generate data that can be interpreted as musical data, we consider that a note is an item, a measure is a sequence and a score is obtained by concatenation of the sequences belonging to the database. The strategy, that we used to control the motifs belonging to the sequence, consists in generating a dataset without motif, then, generating motifs and introduces them into the dataset. SPMF generator has been used to build sequences with large vocabulary (maximum distinct items) and small itemsets (item count per itemset) with the following parameters [16]. Sequence count, maximum distinct items ( $\mathcal{I}$ ), item count by itemset and itemset count per sequence ( $\mathcal{S}$ ) have been fixed respectively to 500, 1000, 1, and 10.

Thus, with these settings, in the generated dataset, called *D500I1KT10*, the probability of having *freq* occurrences of a motif of size  $l$  is very low, estimated to  $(\mathcal{A}_l^{|\mathcal{I}|} / \mathcal{A}_{|\mathcal{S}|}^{|\mathcal{I}|})^{freq}$ .

Then, a set of *contiguous motifs* has been generated as well as a set of gaps which have been inserted into the motifs to form non-contiguous motifs. Both types of motifs have been inserted independently in D500I1KT10 leading to two databases, each containing ten datasets: the datasets in Database1 contain sequences with contiguous motifs whereas those in Database2 contain sequences with non contiguous motifs.

These datasets allow to evaluate CSMA2, the version of our algorithm able to handle a set of sequences. For this reason, CSMA2 has been compared to CM-SPADE, proposed by Philippe Fournier-Viger [17]. As expected, CSMA2 and CM-SPADE have found the same motifs with exactly the same frequencies. The first conclusion is that CSMA is able to extract contiguous and non-contiguous motifs from databases composed of different sequences but one of its advantage is to provide the positions of the detected motifs.

The second part of the evaluation concerns CSMA1, the version of our algorithm which searches motifs from a single sequence. For this experiment, each dataset has been transformed into one string, by concatenating all the sequences together. Thus, we obtain two other databases built from Database 1 and Database 2, called respectively string1, with contiguous motifs, and string2, with non contiguous motifs. The results are evaluated according to the number of identified distinct motifs and the frequency (number of occurrences corresponding to these motifs). The average values  $\mu$  and standard deviations  $\sigma$  computed over the 10 datasets for each database are reported as final results. CSMA1 is compared with the algorithm of Liu. For CSMA, minFrequency, minLength, maxLength have been set respectively to 2, 2, 20 with maxGap equals to 0 for string1 and to 3 for string2. The results are presented in Table 2.

Concerning the datasets containing a sequence without gap (string1), the results show that CSMA1 and Liu algorithm find approximatively the same numbers of distinct motifs with almost the same frequencies. The difference comes from the fact that Liu's algorithm identifies only non-trivial motifs when our algorithm detects all the motifs verifying the constraints. For the datasets with gaps (string2), we observe the same phenomena but the difference is more



**Table 2.** String mining results

Algorithms	String1 ( $\mu(\sigma)$ )		String2 ( $\mu(\sigma)$ )	
	Number of distinct motifs	Frequencies	Number of distinct motifs	Frequencies
CSMA1	1100.6 (41.45)	7893.2(1210.14)	6964.8 (2669.22)	25514.8 (10035,26)
Liu [13]	1003.4(38.94)	7088.4 (869.72)	1073.1 (21)	6891.4 (914)

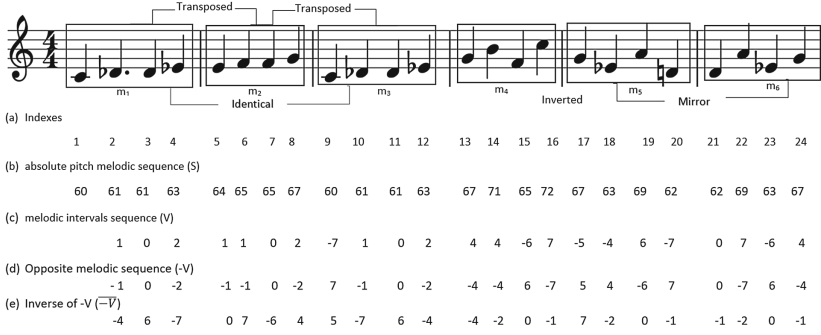
important. This is explained by the fact that long motifs are broken into smaller ones, due to the gaps, but they remain frequent. However, since CSMA1 allows gaps of length 3, the number of motifs detected by CSMA is higher than with Liu algorithm. So, CSMA is able to extract contiguous and non-contiguous motifs from a single sequence and it can take into account the gaps, which is not the case of the Liu’s algorithm.

In conclusion, with its both versions, CSMA can find contiguous and non-contiguous motifs, with or without gaps, from a single string or a set of strings.

## 4.2 Real Music Data

**Preprocessing of the Music Data.** Music symbol-based features are usually represented by pitch values. These values contain two kinds of informations: melodic and rhythmic. Melodic information corresponds to an alphabetic letter, followed by the octave value that can be encoded in MIDI format. For instance, a C note on the 5<sup>th</sup> octave, written C5 corresponds to 72 in MIDI format. The rhythmic information corresponds to the note duration. The aim of our work is to identify musical motifs which can be melodic, rhythmic or both (pitch sequence) and our experiments have shown that CSMA is able to do it. Moreover, we search also three musical variants of a motif: its transposed, inverted or mirror form as illustrated on Fig. 1. A transposed motif is a music part that contains the same tonal variation than an other part in the music score, for instance in Fig. 1,  $m_2$  is a transposed form of  $m_1$ . An inverted motif is a part that is characterized by inverted tonal variation of an other part, like for example,  $m_4$  and  $m_5$ . Finally, a mirror motif corresponds to a symmetric positioning of notes such as  $m_5$  and  $m_6$ . These forms are useful to characterize a composer or a music score. They can be used as a signature for other data mining tasks such as supervised and non supervised clustering or composer identification.

We introduce this section with the presentation of a method based on a preprocessing of the initial melodic sequence  $S$ , able to detect their melodic variants. To this end, we consider the sequence of intervals between consecutive notes from  $S$ . By this way, three new sequences are built. The first one, denoted  $V$  corresponds to the melodic variation between two notes. For instance in Fig. 1, the variation between the note in the first position (60) and the note in the second one (61) equals 1 (61-60). The second sequence, denoted  $-V$  is obtained by taking the opposite of each value in the sequence  $V$ , so each positive value in  $V$  becomes negative and vice-versa. 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. Once the primary and secondary sequences have



**Fig. 1.** musical motifs and their variants

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  $\langle V, -V \rangle$  of size  $2l_v$  on which CSMA is applied. Then, if CSMA generates a new motif having 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 in that positions then we have a *transposed form* at positions  $i$  and  $j$ . For example, as we can see in Fig. 1(c), the motif with the value  $\langle 1 \ 0 \ 2 \rangle$  has three positions which correspond to 1, 5 and 9. As the positions 1 and 9 correspond to the identical motifs  $m_1$  and  $m_3$  previously found, it remains the motif at position 5 which corresponds to  $m_2$ . So we conclude that  $m_2$  is a transposed motif of  $m_1$  and  $m_3$ .

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  $S_1 = \langle S_{j-l_v}, \dots, S_{j-l_v+len_j} \rangle$  is an *inverted form* of the subsequence  $S_2 = \langle S_{i-1}, \dots, S_{i-1+len_i} \rangle$ . For example, when we concatenate sequence (c) and (d) from Fig. 1, the motif with the value  $\langle 4 \ -6 \ 7 \rangle$  is present at position 13, which is lower than 23 ( $l_v$ ), and at position 40, which is greater than 23. Moreover,  $S_{i-1} = S_{13-1} = 67$  and  $S_{j-l_v} = S_{40-23} = S_{17} = 67$ . So the motif  $m_5$ , which starts from position 17, is an inverted form of the motif  $m_4$ , which starts from position 13.

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  $\langle V, -\overline{V} \rangle$  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*.

For example, in Fig. 1(c)(e), the motif with value  $\langle -4 \ 6 \ -7 \rangle$  appears at two positions in  $\langle V, -\overline{V} \rangle$ , one before  $l_v$  and another one after. As the middle value variation is equal to 0, then the motif  $\langle -4 \ 6 \ -7 \ 0 \ 7 \ -6 \ 4 \rangle$  starting from position 17 to 24, that includes  $m_5$  and  $m_6$ , is a mirror motif.

**Table 3.** Number of distinct musical motifs (and variants) in real music scores

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

**Evaluation on 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 3.

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. Even if it has been designed to extract musical motifs, CSMA can be used in other contexts.

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. A software is under designing. It allows to display the extracted motifs on the music sheet and it will be used for an evaluation by the expert. Moreover, in future works, we plan to optimize CSMA in order to improve the running time and to use the motifs and their variants as features representing composers in the context of clustering tasks.

**Acknowledgement.** The funding for this project was provided by a grant from *la région Rhone Alpes*.

## References

1. Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, VLDB, vol. 1215, pp. 487–499 (1994)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14. IEEE (1995)
3. Mooney, C.H., Roddick, J.F.: Sequential pattern mining-approaches and algorithms. *ACM Comput. Surv. (CSUR)* **45**(2), 19 (2013)
4. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C.: Freespan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth ACM SIGKDD, pp. 355–359. ACM (2000)
5. Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th ICDE, pp. 215–224 (2001)
6. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. *Mach. Learn.* **42**(1), 31–60 (2001)
7. Zaki, M.J.: Sequence mining in categorical domains: incorporating constraints. In: Proceedings of the ninth ICIKM, pp. 422–429. ACM (2000)
8. Pei, J., Han, J., Wang, W.: Constraint-based sequential pattern mining: the pattern-growth methods. *J. Intell. Inf. Syst.* **28**(2), 133–160 (2007)
9. Floratou, A., Tata, S., Patel, J.M.: Efficient and accurate discovery of patterns in sequence data sets. *IEEE Trans. Knowl. Data Eng.* **23**(8), 1154–1168 (2011)
10. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. *DMKD* **1**(3), 259–289 (1997)
11. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S.: A survey of sequential pattern mining. *Data Sci. Pattern Recogn.* **1**(1), 54–77 (2017)
12. Hsu, J.L., Chen, A.L., Liu, C.C.: Efficient repeating pattern finding in music databases. In: Proceedings of the seventh ICIKM, pp. 281–288. ACM (1998)
13. Liu, C.C., Hsu, J.L., Chen, A.L.: Efficient theme and non-trivial repeating pattern discovering in music databases. In: Proceedings, 15th International Conference on Data Engineering, pp. 14–21. IEEE (1999)
14. Fuchs, B.: Co-construction interactive de connaissances, application à l’analyse mélodique. In: IC 2011, 22èmes Journées francophones d’Ingénierie des Connaissances, pp. 705–722 (2012)

15. Jiménez, A., Molina-Solana, M., Berzal, F., Fajardo, W.: Mining transposed motifs in music. *J. Intell. Inf. Syst.* **36**(1), 99–115 (2011)
16. Fournier-Viger, P., Lin, J.C.-W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The SPMF open-source data mining library version 2. In: Berendt, B., Bringmann, B., Fromont, É., Garriga, G., Miettinen, P., Tatti, N., Tresp, V. (eds.) *ECML PKDD 2016*. LNCS, vol. 9853, pp. 36–40. Springer, Cham (2016). doi:[10.1007/978-3-319-46131-1\\_8](https://doi.org/10.1007/978-3-319-46131-1_8)
17. Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.: Fast vertical mining of sequential patterns using co-occurrence information. In: Tseng, V.S., Ho, T.B., Zhou, Z.-H., Chen, A.L.P., Kao, H.-Y. (eds.) *PAKDD 2014*. LNCS, vol. 8443, pp. 40–52. Springer, Cham (2014). doi:[10.1007/978-3-319-06608-0\\_4](https://doi.org/10.1007/978-3-319-06608-0_4)

Advances in Intelligent Data Analysis XVI  
16th International Symposium, IDA 2017, London, UK,  
October 26-28, 2017, Proceedings  
Adams, N.; Tucker, A.; Weston, D. (Eds.)  
2017, XIII, 348 p. 96 illus., Softcover  
ISBN: 978-3-319-68764-3