Fundamenta Informaticae XX (2009) 1–16
IOS Press

Toward a General Framework for Polyphonic Comparison *

Julien Allali
LaBRI - Université de Bordeaux 1 F-33405 Talence cedex, France
julien.allali@labri.fr

Pascal Ferraro C
LaBRI - Université de Bordeaux 1 F-33405 Talence cedex, France
and Pacific Institute For the Mathematical Sciences, University of Calgary, Canada
pascal.ferraro@labri.fr

Pierre Hanna
LaBRI - Université de Bordeaux 1 F-33405 Talence cedex, France
pierre.hanna@labri.fr

Costas Iliopoulos
King's College - London, England
costas.iliopoulos@kings.uk

Matthias Robine
LaBRI - Université de Bordeaux 1 F-33405 Talence cedex, France
matthias.robine@labri.fr

Abstract. Existing symbolic music comparison systems generally consider monophonic music or monophonic reduction of polyphonic music. Adaptation of alignment algorithms to music leads to accurate systems, but their extensions to polyphonic music raise new problems. Indeed, a chord may match several consecutive notes, or the difference between two similar motifs may be a few swapped notes. Moreover, the substitution scores between chords are difficult to set up. In this paper, we propose a general framework for polyphonic music using the substitution score scheme set for monophonic music, which allows new operations by extending the operations proposed by Mongeau and Sankoff [15]. From a practical point of view, the limitation of chord sizes and the number of notes that can be merged consecutively lead to a complexity that remains quadratic.

Keywords: music comparison, polyphony, alignment, edit operations, chord matching

Address for correspondence: Pascal Ferraro - LaBRI - Université de Bordeaux 1 - F-33405 Talence cedex, France, `pascal.ferraro@labri.fr`
*This work has been partially sponsored by the French ANR SIMBALS (JC07-188930) and ANR Brasero (ANR-06-BLAN-0045) projects.
CCorresponding author

1. Introduction

One of the main goals of music retrieval systems is to find musical pieces in large databases given a description or an example. These systems compute a numeric score on how well a query matches each piece in a database and rank the music pieces according to this score. Computing such a degree of resemblance between two pieces of music is a difficult problem. Three families of methodologies have been proposed [16]. Approaches based on index terms generally consider N-gram techniques [4, 26], which count the number of common distinct terms between the query and a potential answer. Geometric algorithms [28, 24, 25] consider geometric representations of music and compute distances between objects. Techniques based on string matching [10] can take into account errors in the query or in the pieces of music of the database. This property is of major importance in the context of music retrieval systems since transcription and recognition of audio signal is never completely accurate. Moreover, some music retrieval applications require specific robustness. Query by humming (QbH), a music retrieval system where the input query is a user-hummed melody, is a very good example. Edit-distance algorithms, mainly developed in the context of DNA sequence recognition, have been adapted in the context of music similarity [15]. These algorithms, based on the dynamic programming principle, are generalizations of a local sequence alignment method proposed by Smith and Waterman [20] in the early 80's. Applications relying on local alignment are numerous and include cover detection [19], melody retrieval [15], Query-by-Humming [3], structural analysis, comparison of chord progressions [2], *etc*. Local alignment approaches usually provide very accurate results as shown during the recent editions of the Music Information Retrieval Evaluation eXchange (MIREX) [6].

Symbolic melodic similarity systems generally assume a monophonic context. Therefore, their applications to polyphonic music require accurate adaptations: the monophonic melody has first to be defined and extracted from a polyphonic musical audio signal, and then, the properties of two melodies that induce their similarity to the human mind have to be computed. This second point is closely linked to perception and cognition.

Monophonic music is assumed to be composed of only one dominant melody. In a stricter sense, it implies that no more than one note is sounded at any given time. In music theory, *polyphony* is a texture consisting of two or more independent melodic voices. A music with one dominant melodic voice accompanied by chords is often called *homophony*. However, in the following, we will consider homophony as a part of polyphonic music [27]. Thus, in the polyphonic context, more than one note can sound at a given time. There is no perfect technique to precisely define the melody of such music [27]. Most existing techniques for similarity measurement consider a monophonic context. However, applications of this similarity measurement may concern more complex musical information. For example, one might want to retrieve audio tracks similar to an audio query from a polyphonic audio database. In such applications, the audio example may not be monophonic. In this case, without information about voice lines, most existing approaches would involve transforming polyphonic music into monophonic music [27, 17]. A monophonic sequence is derived from a polyphonic source by selecting at most one note at every time step; this process is called monophonic reduction. In [27], a few existing methods are proposed for monophonic reduction from a polyphonic source. Experiments lead to the conclusion that the perfect technique does not exist, and that each algorithm produces significant errors. Another conclusion is that choosing the highest note of chords yields better results.

Taking into account the approximation induced by the monophonic reduction, we propose here to consider directly polyphonic musical sequences instead of trying to transform polyphonic sources into

monophonic melodies. Such an approach requires a data structure adapted to the constraints induced by the polyphonic context. In particular, the representation has to consider that several notes can sound at the same time.

In this paper, we present a general framework for comparing polyphonic musical pieces by extending the edit-distance operations introduced by Mongeau and Sankoff [15]. In Section 2, we discuss the existing symbolic representations for polyphonic music. Then, in Section 3, we present the existing method for aligning two monophonic musical pieces, and detail the problems due to the polyphonic context. In Section 4, a general algorithm is proposed for aligning two polyphonic musical pieces. Musical examples illustrate the improvements induced by the algorithm in Section 5. Finally, we discuss the limitations and the future work in Section 6.

2. Music Representations and Definitions

In the following, we only consider symbolically encoded pieces of music. Symbolic pieces of music are defined by musical events, such as beginnings or endings of notes. Each note is then defined by a few attributes: pitch, duration or onset time. We will focus in the following on representations of pieces of music as sequences of symbols. Such representations allow the application of algorithms adapted from the string-matching domain. For example, algorithms based on N-grams are proposed in [26, 4]. Another technique evaluates the best alignment between two pieces [20]. In the monophonic context, this method has been experimented as one of the most accurate [10], since it easily allows the consideration of *elements of music theory* such as tonality, passing notes, or strong and weak beats [18].

In order to compare polyphonic music, existing works generally require reduction of a polyphonic piece [26]. Certain methods [13] reduce polyphonic music as a set of separate tracks. We choose to not use any information about the voice lines, since they could be missing in the case of polyphonic music obtained by transcription from audio for example. Other monophonic reductions only consider the note with the highest pitch in the polyphony, based on two assumptions: firstly the query is the main theme or the melody of the musical piece searched, and secondly the highest pitches always form the melody. It is rarely the case in an orchestra, where the polyphony could not be reduced to just the voice of the Western concert flute for example. In order to avoid such assumption, we propose here to study algorithms that take into account all the notes of a polyphonic musical piece. One way would be to consider all the distinct monophonic lines induced by the polyphony. But this naive approach may imply a combinatoric explosion in the cases of large scores [12].

In order to simplify the adaptations of alignment algorithms to the polyphonic context, we use the representation of polyphonic music proposed in [9] as a sequence of sets of symbol pairs. Note that the idea of representing polyphonic music as a sequence of ”simultaneities” has been used for many years [14, 5]. As for monophonic sequence [15], each pair represents a note and is mainly defined with its pitch and its duration. Fig. 1 shows an example of an excerpt of a polyphonic musical piece and its related representation.

According to this representation, we propose to study the algorithm computing the alignment between two polyphonic musical pieces, *i.e.* between two sequences of a set of symbol pairs.

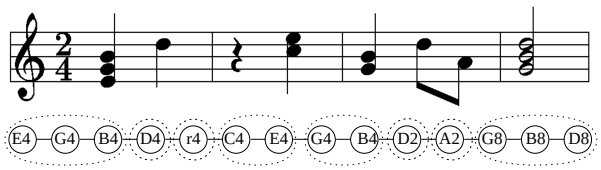

Figure 1. An example of a polyphonic musical score and its related sequence of sets of notes. A note is represented by a pitch and a length, and each note of a same chord belongs to the same set.

3. Aligning Two Pieces of Music

In this section, t and q denote two strings over an alphabet Σ. The cardinality of Σ is $|\Sigma|$ and the lengths of t and q are respectively denoted by $|t|$ and $|q|$. The i^{th} letter of t is represented by t_i, that is $t = t_1 t_2 \dots t_{|t|}$. We note $t_{i\dots j}$ the sub-string made of letters $t_i t_{i+1} \dots t_j$, if $j < i$ then this sub-string is equivalent to the empty word ϵ.

3.1. General Sequence Alignment

Sequence alignment refers to a method that allows the computation of a one-to-one mapping between symbols of two sequences that respects symbol order. Scores are associated with each symbol pair and each symbol not in a pair. Sequence alignment aims at finding a mapping between symbols that maximizes the sum of scores. Here is a formal definition of the problem:

Definition 3.1. (Sequence Alignment)
Let t and q be two sequences over an alphabet Σ, whose respective lengths are $|t|$ and $|q|$. A valid alignment A from t to q is a set of ordered pairs of integers (i, j) satisfying:

1. $1 \leq i \leq |t|$ and $1 \leq j \leq |q|$,
2. for any distinct pairs (i_1, j_1) and (i_2, j_2) in A:
   (a) $i_1 \neq i_2$ if and only if $j_1 \neq j_2$,
   (b) $i_1 < i_2$ if and only if $j_1 < j_2$.

Condition 1 ensures that only character positions of the respective strings are involved in the alignment. Condition 2(a) ensures that each character position of either string is aligned with at most one character position of the other string; condition 2(b) ensures that the order between character positions is maintained in the alignment. Let I and J be the sets of positions in t and q respectively not involved in any pair in A.

Let $score$ be an arbitrary scoring function which assigns to each pair (a, b) of $\Sigma \cup \{\epsilon\} \times \Sigma \cup \{\epsilon\}$ a real number $score(a, b)$. Let A be an alignment from t to q, the score $S(A)$ associated with the alignment A is defined by:

$$S(A) = \sum_{(i,j)\in A} score(t_i, q_j) + \sum_{i\in I} score(t_i, \epsilon) + \sum_{j\in J} score(\epsilon, q_j).$$

Finally, the alignment score $align^t(t, q)$ between t and q is defined as:

$$align^t(t, q) = Max(S(A)\ for\ all\ valid\ A\ between\ t\ and\ q).$$

A pair of two identical symbols in the alignment A is called a *match*. A pair of two different symbols is called a *mismatch*. Symbols not involved in the alignment are called *gaps*. Thus the score of A is the cost of the match and mismatch pairs, added to the cost of gaps.

The alignment A from t to q is usually represented by two sequences t' and q' of the same length over the alphabet $\Sigma \cup \{-\}$. t and q are respectively obtained by suppressing the symbols $-$ from respectively t' and q'. Let us consider the following alignment from the word $MUSIC$ to the word $QUICK$:

position:	0	1	2	3	4	5
	M	U	S	I	C	-
	Q	U	-	I	C	K
score:	-1	2	-1	2	2	-1

In this example, we have a mismatch in position 0, a match in positions 1, 3 and 4 and two gaps in positions 2 and 5. If the score for a match is +2 and -1 for a mismatch or a gap then the alignment score is 3.

The alignment score $align^t(t, q)$ from t to q may be computed rapidly (in time proportional to $|t| \times |q|$ by recurrence). The recurrence equation from $1 \leq i \leq |t|$ and $1 \leq j \leq |q|$ is (see Fig. 2-left):

$$align^t(t_{1\ldots i}, q_{1\ldots j}) = Max \begin{cases} align^t(t_{1\ldots i-1}, q_{1\ldots j}) & + & score(t_i, \epsilon) \\ align^t(t_{1\ldots i}, q_{1\ldots j-1}) & + & score(\epsilon, q_j) \\ align^t(t_{1\ldots i-1}, q_{1\ldots j-1}) & + & score(t_i, q_j) \end{cases} \quad (1)$$

The computation can be done by computing a table M of size $(|t| + 1) \times (|q| + 1)$ where the value $M[i][j]$ represents $align^t(t_{1\ldots i}, q_{1\ldots j})$. The first row (resp. column) contains scores of the alignment between $t_{1\ldots i}$ (resp. $q_{1\ldots j}$) and ϵ. The time complexity of this algorithm is $O(|t| \times |q|)$ in time and $O(min(|t|, |q|))$ in memory as we can make the computation using only one row or column [20].

Several variants of the global alignment definition (Def. 1) have been proposed in the literature [8]. The most well known are:

- *local alignment:* finding factors of q and t having the best alignment score
- *best fit:* finding the factor of t that has the best alignment score with q.

These variants are computed in a similar way to global alignment and the present work can be easily adapted to local and best-fit polyphonic alignment.

When comparing monophonic pieces of music one limitation of the alignment approach is that it only allows one-to-one association. Actually, in musical pieces, a single note in one sequence may sometimes be split into two or more notes in the second sequence. To avoid this limitation, Mongeau and Sankoff [15] introduced a new operation, called *merge* allowing the replacement of several characters by a single character and the replacement of one character by several. The motivation for this new operation, is the fact that a whole note can be replaced by four quarter notes of the same pitch, which is not very costly.

The definition of an alignment is then modified as follows:

Definition 3.2. Let t and q be two sequences over an alphabet Σ, whose respective lengths are $|t|$ and $|q|$. A valid alignment A from t to q is a set of pairs of ordered pairs of integers $((i_s, i_e), (j_s, j_e))$ satisfying:

1. $1 \leq i_s, i_e \leq |t|$ and $1 \leq j_s, j_e \leq |q|$; and $i_e > i_s$ and $j_e > j_s$,
2. $i_e \neq i_s + 1 \Rightarrow j_e = j_s + 1$ and $j_e \neq j_s + 1 \Rightarrow i_e = i_s + 1$,
3. if $((i'_s, i'_e), (j'_s, j'_e))$ and $((i'_s, i'_e), (j'_s, j'_e))$ are any two distinct elements of A:
   (a) i'_s cannot be equal to i_s,
   (b) if $i'_s < i_s$ then $i'_e \leq i_s$ and $j'_e \leq j_s$,
   (c) if $i'_s > i_s$ then $i'_s \geq i_e$ and $j'_s \geq j_e$.

Condition 1 ensures that only character positions of the respective strings are involved by the alignment; Condition 2 ensures that a sequence of two non consecutive character positions in a string is associated with at most one character position; Conditions 3 ensure that the order between character position is maintained in the alignment.

To take into account the merge operation in the recurrence 1, the following cases must be added to the recurrence formula (see Fig. 2-right):

$$\begin{array}{lll} 2 & \leq k \leq i: & align^t(t_{1\ldots i-k}, q_{1\ldots j-1}) + score(t_{i-k\ldots i}, q_j), \\ 2 & \leq l \leq j: & align^t(t_{1\ldots i-1}, q_{1\ldots j-l}) + score(t_i, q_{j-l\ldots j}). \end{array} \tag{2}$$

where $score(t_{i-k\ldots i}, q_j)$ and $score(t_i, q_{j-l\ldots j})$ are the predefined weights associated to the merge operations.

The time complexity of the algorithm with the merge operation is $O((|t| \times |q|)(|t| + |q|))$. In practice, the number of consecutive merged notes is bounded by a constant L which leads to a complexity of $O(|t| \times |q| \times L)$ [15].

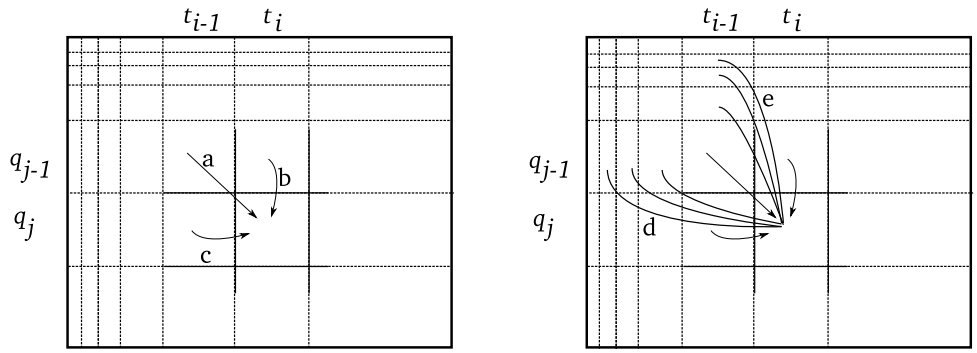

Figure 2. On the left, the dynamic programming table for the alignment of t and q. The arrows a, b and c represent the three cases of recurrence 1 for the computation of $M[i+1][j+1]$. On the right, the computations added by Mongeau and Sankoff algorithm (Formula 2). Arrows d correspond to merge in t, arrows e correspond to merge in q.

So far, to define an accurate algorithm, a scoring function must be defined. A naive scoring scheme like constant operation scores or a complex scoring scheme can produce significantly different results [10]. Setting these parameters is tricky and generally leads to a scoring table that contains the scores

between each possible pitch difference. Indeed, substituting one pitch with another one has more or less influence on the general melody. As introduced in [15], scores are thus determined according to consonance. In particular, interval scores decrease for Western tonal music in order of decreasing dissonance: unison, fifth, third, sixth, fourth, seventh and second.

In the monophonic case, a good scoring scheme between two notes t_i and q_j is often reached by using a function $score_p$ on pitches (its values are coded into a table) and a function $score_d$ on durations, such as:

$$score(t_i, q_j) = \alpha \times score_p(pitch(t_i), pitch(q_j)) \\ +\beta \times score_d(duration(t_i), duration(q_j)).$$

For a merge operation the weight $score(t_{i-k...i}, q_j)$ is computed similarly as:

$$score(t_{k...i}, q_j) = \alpha \times \sum_{l=k}^{i} score_p(pitch(t_l), pitch(q_j)) \\ +\beta \times score_d(\sum_{l=k}^{i} duration(t_l), duration(q_j)).$$

Usually, the cost associated to a gap only depends on the note duration.

3.2. Polyphonic Case

A lot of problems arise, when dealing with polyphonic music alignment. Actually, the definition of an alignment in the polyphonic case is not a straightforward application of the monophonic comparison.

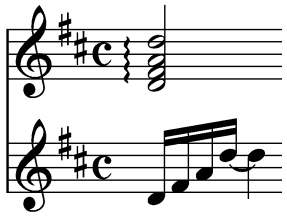

Figure 3. Arpeggiated chord above, with its interpretation below. The notes are played successively from the lowest to the highest pitch, and not simultaneously.

Since many notes may be played at the same time, relative encoding cannot be used. Thus, polyphonic music can be represented by a sequence of sets of notes. A set can contain a single note or a chord. A direct consequence is that transpositions cannot be treated by the relative encoding. This problem has been addressed in [1] and requires computing simultaneously multiple matrices, one for each possible transposition value.

Furthermore, as presented before, setting up a scoring scheme in the monophonic case (*i.e.* fixing a score for two notes) is a difficult problem. This task becomes harder when comparing two chords. Indeed, in one octave there are 12 possible pitch values for a chord made of a single note (in practical

applications, it is common to work only on one octave), then 12×11 for two note chords $\ldots \binom{12}{p}$ for p note chords, which means the scoring scheme will be represented by a matrix of size $2^{12} \times 2^{12}$.

Moreover, complex note rearrangements may occur between two similar polyphonic pieces, and temporal deviations may appear between a score and its interpretation. Fig. 3 shows such an example: in the notation score, the chord has to be arpeggiated, and the related interpretation is transcribed as successive notes. In this case, a comparison system has to detect the similarity between the chord indicated in the musical score and the successive notes interpreted.

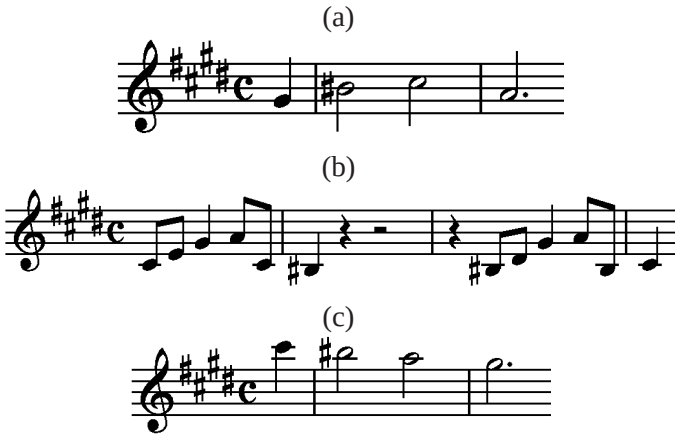

Figure 4. Similarity despite permutations (a) Main motif of the 14th string quartet in C# minor opus 131 by Beethoven (1st movement, bar 1). The motif is composed by 4 notes (sequence (1 2 3 4)). (b) First theme of the 7^{th} movement, bar 1. The 4 last notes of the two groups are permuted notes of the main motif, sequence (1 4 3 2) and (1 4 2 3) (c) Second theme of the 7^{th} movement, bar 21. The 4 notes are again a permutation of the main motif, sequence (3 2 4 1).

More generally, we have to deal with notes/chords merging and local rearrangements. For example, composers may choose to change the order of the notes in a melodic motif during a musical piece. Fig. 4 shows three excerpts from a piece by Beethoven: the second (b) and third (c) excerpts correspond to a rearrangement of the main motif (a) with swapped notes.

4. Polyphonic Alignment

We propose hereafter a general algorithm to align two polyphonic musical pieces. The main characteristic of this algorithm is that it is based on a scoring scheme in monophonic music.

4.1. Chord Comparison

In many cases, an arbitrary order is given to the notes composing the chords of a musical sequence. To avoid this arbitrary choice, one can consider chords as sets. The cost for substituting one chord by another one leads to the problem of computing the best permutation between both chords. Fig. 5 shows an example of two cadences that sound similar, but that can be estimated as very dissimilar because of the different order of the notes in the chords. To avoid this sort of problem, we suggest that chords should

be considered as unordered sets and the best permutation should be found. This optimization method allows the estimation of a high similarity between these two sequences of chords.

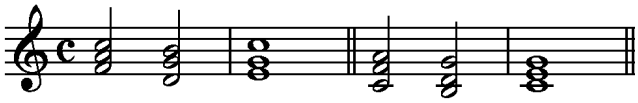

Figure 5. Similarity between inverted chords. These successive two perfect cadences in C major are similar despite the different order of the notes in the chords composing the cadences.

This optimization problem is actually a maximum score maximum bipartite matching problem and can be modeled as a weighted maximum matching algorithm [7]. A similar approach has been proposed in the case of a geometric representation of musical pieces [23].

Let C_1 and C_2 be two chords of size n and m. We denote by $score_{bpg}(C_1, C_2)$ the score between these two chords. To compute $score_{bpg}(C_1, C_2)$ as the maximum score maximum bipartite matching problem, we consider the following graph $G(v, w) = (V, E)$ (Fig. 6):

1. *vertex set* : $V = \{s, t, e_1, e_2\} \cup \{s_1^1, s_1^2, \dots s_1^n\} \cup \{s_2^1, s_2^2, \dots s_2^m\}$, where s is the source, t is the sink, $\{s_1^1, s_1^2, \dots s_1^n\}$ and $\{s_2^1, s_2^2, \dots s_2^m\}$ are the notes of the chords C_1 and C_2 and e_1, e_2 represent ϵ;

2. *edge set* : (s, s_1^k), (s, e_1), (e_2, t), (s_2^l, t) with a score 0, (s_1^k, s_2^l) with score $score(s_1^k, s_2^l)$, and (s_1^k, e) with score $score(s_1^k, \epsilon)$. All the edges have a capacity of 1 except (e, t) which capacity is $n - m$.

G is then a graph whose edges are labeled with integer capacities, non-negative scores in $\mathbb{R}$, and the maximum flow $f^* = n + m$. The score of the maximum flow is actually the score $score_{bpg}(C_1, C_2)$ and the complexity of computing local score is due to this maximum score maximum flow computation. This problem can be solved by the Edmonds and Karp's algorithm [7] improved by Tarjan [21] whose complexity is $O(|E||f^*| \log_2(|V|))$. For our graph, the maximum flow is $f^* = n + m$, the number of edges is $|E| = n \times m + 2n + 2m + 3$ and the number of vertices is $|V| = n + m + 4$. Finally the complexity of the score computation between two chords is bounded by $O(n^3 \times \log_2(n))$ where n represents the maximum number of notes in a chord.

In conclusion, computing alignment between two strings t and q leads to a total time complexity of $O(|t| \times |q| \times C^3 \times \log_2(C))$ where C is the maximum number of notes in a chord in t or q. In practical applications the parameter C is generally bounded by 4.

4.2. Extending Mongeau-Sankoff Operations

As we already stated, an accurate algorithm for music alignment must take into account both local rearrangements and merging operations. We thus propose allowing the merging of sub-sequences in both q and t simultaneously.

Music alignment between two sequences is then extended as follows:

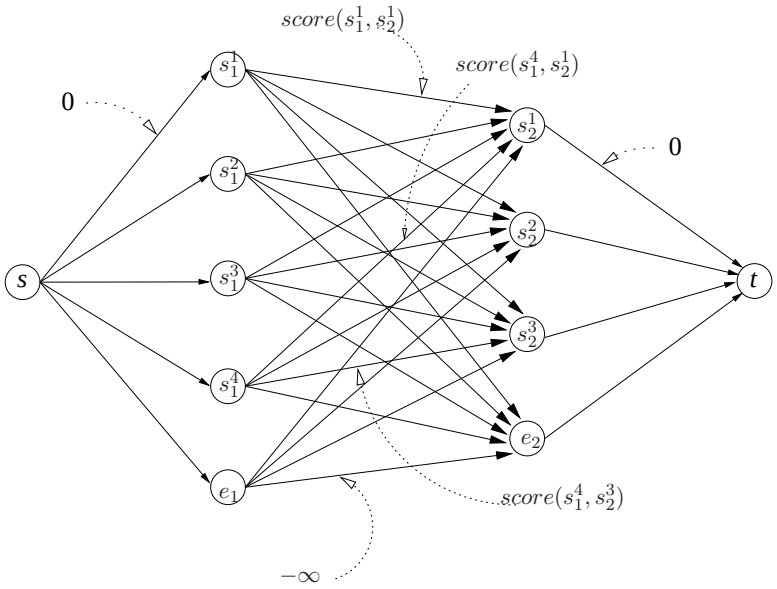

Figure 6. Resolution of the optimal permutation as a maximum score flow problem.

Definition 4.1. (Extended Music Alignment)
Given two sequences t and q of sets of symbols over Σ. A valid extended alignment X^a between t and q is a set of pairs of sub-sequences of t and q, that is $X^a = \{((i_s, i_e), (j_s, j_e))\}$ and respects the following for all $((i_s, i_e), (j_s, j_e)) \in X^a$:

1. $1 \leq i_s < i_e \leq |t|$ and $1 \leq j_s < j_e \leq |q|$,
2. $\forall((i'_s, i'_e), (j'_s, j'_e)), ((i'_s, i'_e), (j'_s, j'_e)) \in X^a$ such that $i'_s \neq i_s$:
   (a) if $i'_s < i_s$ then $i_e \leq i_s$ and $j'_e \leq j_s$,
   (b) if $i'_s > i_s$ then $i'_s \geq i_e$ and $j'_s \geq j_e$.

We define the set G_t (resp. G_q) as the set of positions of t (resp. q) not involved in X^a, that is

$$G_t = \{1, \ldots, |t|\} \backslash \bigcup_{((i_s,i_e),(j_s,j_e))\in X^a} \{i_s \ldots i_e\}.$$

The score associated with X^a is defined as follows:

$$S(X^a) = \sum_{((i_s,i_e),(j_s,j_e))\in X^a} score(t_{i_s \ldots i_e}, q_{j_s \ldots j_e}) + \sum_{i \in G_t} score(t_i, \epsilon) + \sum_{j \in G_q} score(\epsilon, q_j).$$

The adaptation of the alignment algorithm is straightforward. The recurrence 1 is modified by adding the following cases:

$$\forall (k, l) \in \{2..i\} \times \{2..j\} : align^t(t_{1 \ldots i-k}, q_{1 \ldots j-l}) + score(t_{i-k \ldots i}, q_{j-l \ldots j}). \quad (3)$$

In the dynamic programming table, the value of a given position $M[i][j]$ is obtained from any case $M[k][l]$ for $1 \leq k \leq i$, $1 \leq l \leq j$. We thus need to consider the computation of the value $score(t_{i-k...i}, q_{j-l...j})$ (*ie.* the cost of aligning the sequence of chords $t_{i-k...i}$ and the sequence of chords $q_{j-l...j}$).

For the comparison of two polyphonic sequences t and q, we propose to define a scoring scheme based on the scoring scheme presented in Section 3.1. However, in this case, we are dealing with sequences of chords instead of single chords and we thus need a way to encode several chords into a single one.

Let us consider the sequence of chords $t_{i_s...i_e}$ that must be encoded (and that will be compared to a sequence of chords $q_{j_s...j_e}$ in q). The pitch of $t_{i_s...i_e}$ is then defined as the set T_p of all the different pitches that are present in this sequence. The duration of this sequence of chords is equal to T_d the duration elapsed from the beginning of t_{i_s} to the end of t_{i_e}.

For example, in Fig. 1, the sequence of 4 chords:

$$(\{G4, B4\}, \{D2\}, \{A2\}, \{G8, B8, D8\})$$

is encoded by the chord:

$$(\{A, B, D, G\})$$

with a duration of 16.

Finally, a sequence of chords is only represented by a single set of unordered pitches and a single duration. To compare these chord representations the approach described in Section 4.1 and $score_p$ to weight the edges of the bipartite graph are used. Then, the score between two sequences of chords is given by:

$$score(t_{i_s...i_e}, q_{j_s...j_e}) = \alpha \times score_{bpg}(T_p, Q_p) + \beta \times score_d(T_p, Q_d)$$

		{A1}	{C2,G2}	{C1,E1,B1}
	...	...	...	...
{G4,B4 }	...	a	b	
{D2}	...	c	d	
{A2}	...	e	f	
{G8,D8,B8}	...			X

Table 1. Example of merged notes that must be considered for one step (X) of the alignment algorithm.

Now let us consider the computation of the dynamic programming table M. We propose to illustrate the computation of a case $M[i][j]$ of this table through the example in Tab. 1. The score in position X is obtained either from f which implies the computation of scores $score_{bpg}(\{A\}, \{C, G\})$ and $score_d(2, 2)$ or:

- from e which implies the computation of $score_{bpg}(\{A\}, \{A, C, G\})$ and $score_d(2, 3)$.
- from d which implies the computation of $score_{bpg}(\{A, D\}, \{C, G\})$ and $score_d(4, 2)$.

- from c which implies the computation of $score_{bpg}(\{A, D\}, \{A, C, G\})$ and $score_d(4, 3)$.
- from b which implies the computation of $score_{bpg}(\{A, B, D, G\}, \{C, G\})$ and $score_d(8, 2)$.
- from a which implies the computation of $score_{bpg}(\{A, B, D, G\}, \{A, C, G\})$ and $score_d(8, 3)$.

One can observe that from one computation of $score_{bpg}$ to another one, we just add vertices in the bipartite graph. So, it is not necessary to recompute $score_{bpg}$ from scratch. Toroslu and Üçoluk give in [22] an incremental algorithm to compute the assignment problem in $O(V^2)$ where V is the number of vertices in the bipartite graph. Using this algorithm in our approach the time complexity of the computation of all possible merges for the case i, j is bounded by $O(\sum_{i=1}^{C} i^2) = O(C^3)$ where C is number of different pitches in $t_{1...i}$ and $q_{1...j}$. The time complexity of the alignment becomes $O(|t|^2 \times |q|^2 \times C^3)$ where C is the number of different pitches in t and q.

5. Experimentation and Results

5.1. An Illustrative Example

In order to illustrate the algorithm, we propose to compare the following sequence of chords: t =({G4}, {B4},{D2,A2},{G8},{D8},{B8}) (*cf.* Fig. 1) and an arpeggiated version of this sequence: q =({G4,B4}, {D2},{A2},{G8,D8,B8}). The distance between t and q is computed using the following scoring scheme. Let us consider two notes a and b:

- $score(a, a) = 2$,
- $score(a, b) = 1$ if $a \neq b$,
- $score(a, \epsilon) = score(\epsilon, b) = -1$,

In this example, we suppose the score between two notes does not depend on the pitch and the duration of the notes. The dynamic programing table computation is represented in Tab. 2. We then can observe, for instance, that the score between the subsets of notes ({G4},{B4},{D2,A2}) and ({G4,B4},{D2},{A2}), is obtained from the merge of chords ({G4},{B4}) and ({G4,B4}) on one hand and on the other hand from the merge of the chords ({D2,A2}) and ({D2},{A2}). This score (*i.e.* 8) is actually computed by summing the score of the permutation between {D2,A2} and {D2,A2} (*i.e.* 4) and the score between the chords ({G4},{B4}) and ({G4,B4}) (*i.e.* 4).

The final score between the two global sequences of chords t and q can be obtained either by merging all the chords of t to the chords of q (which corresponds to the cost of the optimal bipartite matching between ({G4,B4,D2,A2,G8,D8,B8}) and ({G4,B4,D2,A2,G8,D8,B8}); or from the cost described previously (between ({G4},{B4},{D2,A2}) and ({G4,B4},{D2},{A2}) for which the cost is 8) plus the cost of merging ({G8,D8,B8}) and ({G8},{D8},{B8}).

5.2. A First Evaluation

The evaluation of similarity measure systems is a very difficult task because different human listeners do not always have the same opinion on similarity between musical pieces. However, this section presents a preliminary evaluation of the general framework for polyphonic comparison.

	ϵ	{G4}	{B4}	{D2,A2}	{G8}	{D8}	{B8}
ϵ	0	-1	-2	-4	-5	-6	-7
{G4,B4 }	-2	1	4	2	1	0	-1
{D2}	-3	0	3	5	4	3	2
{A2}	-4	-1	2	8	7	6	5
{G8,D8,B8}	-7	-4	-1	5	8	11	14

Table 2. Example of a dynamic programming table for the comparison between the sequence of chords ({G4},{B4},{D2,A2},{G8},{D8},{B8}) and ({G4,B4},{D2},{A2},{G8,D8,B8})

During MIREX 2006[1], the second task of the symbolic melodic similarity contest consisted in retrieving the most similar pieces from mostly polyphonic collections given a monophonic query. Note that our approach is more general since it allows comparing two polyphonic musical pieces.

Two collections were considered, and 11 queries (hummed or whistled) were proposed. The *mixed* collection is composed of 10000 randomly picked MIDI files that were harvested from the Web and which include different genres. The *karaoke* collection is composed of about 1000 `.kar` files (Karaoke MIDI files) with mostly Western popular music. Tab. 3 presents the results obtained with these two collections and analyzed using our general polyphonic alignment algorithm and the adaptation of the Smith and Waterman algorithm [20] to music applications [10]. Algorithms have been evaluated according to two measures: the *average precision*, and the *precision at N documents* (N is the number of relevant documents).

Collection		**General Polyphonic Framework**	Classical Alignment
Karaoke	AP	**0.78**	0.36
	PND	**0.83**	0.33
Mixed	AP	**0.67**	0.52
	PND	**0.66**	0.55

Table 3. Average Precision (AP) and Precision at N Documents (PND) obtained by edit-distance based retrieval systems for MIREX 2006 databases and queries. The Classical Alignment column presents the results obtained by Uitdenbogerd during MIREX).

Results[2] presented in Tab. 3 clearly show that the algorithm allowing multiple transpositions and several types of edit operations improves retrieval systems based on the alignment principle. Concerning the *karaoke* collection, the average precision is near 0.80 whereas it is only 0.36 when considering a monophonic alignment. This difference (although less pronounced) is also observed for the *mixed* collection. The average precision is 0.67 instead of 0.52. Although voices are not separated in the musical pieces, this improvement remains notable.

[1] `http://www.music-ir.org/mirex2006/index.php/MIREX2006_Results`

[2] The complete results obtained by the different methods proposed during MIREX 2006 can be found at `http://www.music-ir.org/mirex/2006/index.php/Symbolic_Melodic_Similarity_Results`

In order to confirm these first promising results, other experimentations have to be performed with a more important number of pieces and melodies in our collection.

6. Conclusion

In this paper, we proposed a general alignment algorithm for the comparison of two polyphonic pieces of music. This method is based on an extension proposed by Mongeau and Sankoff [15], in order to compare two monophonic musical pieces. In particular, this algorithm only requires a scoring scheme between notes (and not chords). Although new edit operations have been introduced, the algorithm remains quadratic, if the number of different pitches in a chord and the number of merged chords are bounded by constants.

The string-to-string alignment problem consists of determining the distance between two strings as measured by the minimum cost sequence of edit-operations needed to change one string into the other. The edit operations generally allow changing *one symbol* of a string into *another single symbol*, or inserting a *single symbol* or deleting *a single symbol*. Due to our domain of application, we have firstly generalized this notion to the comparison of sequences of sets of symbols: an edit operation allows modifying a set of symbols. While the cost of symbol comparison was stored in an *alignment table*, now to compare two sets of symbols, a maximum score maximum matching problem must be solved.

Another key element introduced in this paper is the notion of merge also known as consolidation and fragmentation operations [15]. This operation is different from the deletions and insertions familiar in sequence comparison and from the compression and expansions of time warping in automatic speech recognition [11]. This new transformation involves the replacement of several elements of the initial sequence by several elements of the final sequence. However, although this new definition differs from the one proposed by Mongeau and Sankoff and increases the combinatorics, it allows us to compute a more accurate similarity measure between two *polyphonic musical pieces* in polynomial time.

Applications of this algorithm could be extended to any kind of sequences in which time is essentially a quantitative dimension (rhythm, in music) and not simply an ordinal parameter. However, further studies might be carried out in a more general framework in order to evaluate the proper accuracy of this approach.

7. ACKNOWLEDGMENT

The authors wish to thank the reviewers for their comments that help to improve the manuscript and Steve Longay for insightful remarks and editorial help.

References

[1] Allali, J., Ferraro, P., Hanna, P., Iliopoulos, C.: Local Transpositions in Alignment of Polyphonic Musical Sequences, *14th String Processing and Information Retrieval Symposium* (N. Ziviani, R. Baeza-Yates, Eds.), 4726, Springer, oct. 2007.

[2] Bello, J.: Audio-based Cover Song Retrieval using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats, *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, September 2007.

[3] Dannenberg, R. B., Birmingham, W. P., Pardo, B., Hu, N., Meek, C., Tzanetakis, G.: A Comparative Evaluation of Search Techniques for Query-by-Humming Using the MUSART Testbed, *Journal of the American Society for Information Science and Technology (JASIST)*, **58**(5), 2007, 687–701.

[4] Doraisamy, S., Rüger, S.: Robust Polyphonic Music Retrieval with N-grams, *Journal of Intelligent Information Systems*, **21**(1), 2003, 53–70, ISSN 0925-9902.

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

[6] Downie, J. S., Bay, M., Ehmann, A. F., Jones, M. C.: Audio Cover Song Identification: MIREX 2006-2007 Results and Analyses, *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08)*, September 14-18 2008.

[7] Edmonds, J., Karp, R. M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, *Journal of the Association for Computing Machinery*, **19**, 1972, 248–264.

[8] Gusfield, D.: *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press, January 1997, ISBN 0521585198.

[9] Hanna, P., Ferraro, P.: Polyphonic Music Retrieval by Local Edition of Quotiented Sequences, *Proceedings of the 5th International Workshop on Content-Based Multimedia Indexing (CBMI)*, Bordeaux, France, 2007.

[10] Hanna, P., Ferraro, P., Robine, M.: On Optimizing the Editing Algorithms for Evaluating Similarity Between Monophonic Musical Sequences, *Journal of New Music Research*, **36**(4), 2007, 267–279.

[11] Kruskal, J. B.: An orverview of sequence comparison, in: *Time Wraps, Strings Edits, and Macromolecules: the theory and practice of sequence comparison* (D. Sankoff, J. B. Kruskal, Eds.), chapter 1, Addison-Wesley Publishing Company Inc, University of Montreal, Montreal, Quebec, Canada, 1983, 1–44.

[12] Lemström, K., Pienimäki, A.: Approaches for Content-Based Retrieval of Symbolically Encoded Polyphonic Music, *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC)*, Bologna, Italy, 2006.

[13] Madsen, S., Typke, R., Widmer, G.: Automatic Reduction of MIDI Files Preserving Relevant Musical Content, *Proceedings of the 6th International Workshop on Adaptive Multimedia Retrieval (AMR'08)*, Berlin, 2008.

[14] Meredith, D.: Computer-aided comparison of syntax systems in three piano pieces by Debussy, *Contemporary Music Review*, **9**(1-2), 1993, 285–304.

[15] Mongeau, M., Sankoff, D.: Comparison of Musical Sequences, *Computers and the Humanities*, **24**(3), 1990, 161–175.

[16] Orio, N.: Music Retrieval: A Tutorial and Review, *Foundations and Trends in Information Retrieval*, **1**(1), 2006, 1–90.

[17] Paiva, R. P., Mendes, T., Cardoso, A.: On the Detection of Melody Notes in Polyphonic Audio, *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, UK, September 2005.

[18] Robine, M., Hanna, P., Ferraro, P.: Music Similarity: Improvements of Edit-based Algorithms by Considering Music Theory, *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval (MIR)*, Augsburg, Germany, 2007.

[19] Serrà, J., Gómez, E.: A Cover Song Identification System Based on Sequences of Tonal Descriptors, *Proceedings of 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007.

[20] Smith, T., Waterman, M.: Identification of Common Molecular Subsequences, *Journal of Molecular Biology*, **147**, 1981, 195–197.

[21] Tarjan, R. E.: *Data Structures and Network Algorithms*, CBMS-NFS - Regional Conference Series In Applied Mathematics, 1983.

[22] Toroslu, I. H., Üçoluk, G.: Incremental Assignment Problem, *Information Sciences*, **177**(6), 2007, 1523–1529, ISSN 0020-0255.

[23] Typke, R.: *Music Retrieval based on Melodic Similarity*, Ph.D. Thesis, Utrecht University, 2007.

[24] Typke, R., Veltkamp, R. C., Wiering, F.: Searching Notated Polyphonic Music Using Transportation Distances, *Proceedings of the 12th ACM Multimedia Conference (MM)*, New-York, USA, 2004.

[25] Typke, R., Walczak-Typke, A.: A Tunneling-Vantage Indexing Method for Non-Metrics, *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, USA, Sept. 2008.

[26] Uitdenbogerd, A. L.: *Music Information Retrieval Technology*, Ph.D. Thesis, RMIT University, Melbourne, Australia, July 2002.

[27] Uitdenbogerd, A. L., Zobel, J.: Manipulation of Music for Melody Matching, *Proceedings of the Sixth ACM International Conference on Multimedia*, 1998.

[28] Ukkonen, E., Lemström, K., Mäkinen, V.: Geometric Algorithms for Transposition Invariant Content-Based Music Retrieval, *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, Baltimore, USA, October 2003.