EXTENDING ALIGNMENT ALGORITHM FOR POLYPHONIC COMPARISON

Julien Allali, Pierre Hanna and Matthias Robine

LaBRI - Universite de Bordeaux 1
F-33405 Talence cedex, France
firstname.name@labri.fr

Pascal Ferraro

Pacific Institute For the Mathematical Sciences
University of Calgary, Canada
firstname.name@ucalgary.ca

ABSTRACT

Existing symbolic music comparison systems generally consider monophonic music or monophonic reduction of polyphonic music. Adaptation of alignment algorithms to the music leads to accurate systems, but extensions to polyphonic music arise new problems. Indeed, a chord may match several notes, or the difference between two similar motifs may be a few swapped notes. Moreover, it is difficult to set up the substitution scores between chords. In this paper, we propose a general framework for polyphonic music which permits to directly apply the substitution score scheme set for monophonic music, and which allows new operations by extending the operations proposed by Mongeau and Sankoff [1]. From a practical point of view, the limitations of the size of chords and the number of notes that can be merged lead to a complexity that remains quadratic.

1. INTRODUCTION

Automatically estimating the similarity between musical pieces is one of the major open problems in music information retrieval research area. One of the applications is content-based music retrieval, that consists of searching a musical piece in a database given a short excerpt of music (query) [2]. Queries may be perfect, for example an *exact* excerpt of a piece. But it may also be *inexact*, for example hummed or whistled, leading to slight time or pitch deviations [3].

In this paper, we only consider symbolically encoded music. Symbolic music is defined by musical events, such as beginnings or endings of notes for example. Each note is then defined by a few attributes such as pitch, duration or onset time.

Different representations for music and related algorithms for comparison have been proposed. Geometric algorithms consider geometric representations of music and compute the distance between objects [4]. Other representations consider music as sequence of symbols. Such representations allow the application of algorithms adapted from string matching domain by computing the best alignment between two pieces [5].

In order to compare polyphonic music, existing works generally require a monophonic reduction of a polyphonic piece [6], for instance by considering the note with the highest pitch. In order to avoid such assumption, we propose here to study algorithms that take into account all the notes of a polyphonic musical piece.

Adaptations of alignment algorithms are difficult because the representation applied for monophonic music induces several problems, in particular setting substitution scores that may become very complex. In order to simplify algorithms, a new representation of polyphonic music has been proposed in [7]. Polyphonic pieces are represented by a sequence of sets of symbol pairs. Fig. 1 shows an example of an excerpt of polyphonic music and its related representation. According to this representation, we propose to study an algorithm computing the alignment between two polyphonic musical pieces, *i.e.* between two sequences of sets.

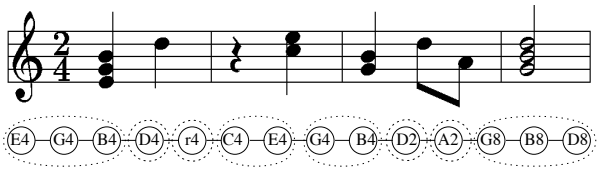

Figure 1: Example of polyphonic musical score and its related sequence of sets of notes. A note is represented by a pitch and a length, and each note of a same chord belongs to the same set.

2. ALIGNING TWO MUSICS

2.1. General Sequence Alignment

Sequence alignment refers to a method that allows the computation of a one-to-one mapping between symbols of two sequences t and q that respects symbol order. Sequence alignment aims at finding a mapping between symbols that maximizes the sum of scores. A pair of two identical symbols in the mapping is called a *match*. A pair of two different symbols is called a *mismatch*. Symbols not involved into the mapping are called *gaps*. Scores, respectively denoted by $\sigma(t_i, q_j)$ and $s(s_i, \epsilon)$, are assigned to each pair (t_i, q_j) of the mapping and gaps. Most famous variant of global alignment (*i.e.* of the whole strings) are *local alignment*, which finds factors of q and t having the best alignment score, and *best fit*, which finds the factor of t having the best alignment score with q [8]. 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. A dynamic programming algorithm allows computing the optimal alignment and the corresponding score, denoted by $score(t, q)$, between two sequences t and q of respective size $|t|$ and $|q|$ in $O(|t| \times |q|)$ time complexity and in $O(min(|t|, |q|))$ in memory complexity [5].

When comparing monophonic music one limitation of alignment approach is that it only allows one-to-one association. However in musical piece 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 [1] introduced the possibility to associate more than one note from one sequence to exactly one note from the other sequence. We call this operation a *merge*. The time complexity of the algorithm with the merge operation is $O((|t| \times |q|)(|t| + |q|))$. In practice, the number of consecutive notes is bounded by a constant L which leads to a complexity of $O(|t| \times |q| \times L)$ [1].

2.2. Polyphonic Case

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

Since many notes may be played at the same time, relative

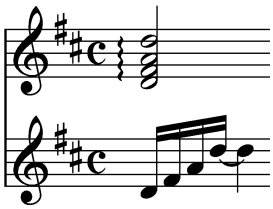

Figure 2: Arpeggiated chord above, with its interpretation below. The notes are played successively from the lowest to the highest pitch, and not simultaneously.

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 [9] and requires to compute simultaneously multiple matrices, one for each possible transposition value.

Furthermore, setting up scoring scheme in monophonic case (*i.e.* fixing a score for two notes) is a difficult problem. This task becomes harder when comparing two chords. Indeed, on 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 or $\binom{12}{p}$ for p note chords, which means the scoring scheme will be represented by a matrix of size $2^{12} \times 2^{12}$.

Lastly, polyphony brings some new complex note rearrangements. The most obvious example is when the notes of a chord are played successively and not simultaneously. It is generally the case when precisely analyzing the interpretation of a musical piece. The automatic transcription may be made of separate notes instead of simultaneous notes. Fig. 2 shows such an example. In the notation score, the chord has to be arpeggiated: the related interpretation is transcribed as successive notes. In this case, a comparison system has to detect the similarity between the chords indicated in the musical score and the successive notes interpreted.

More generally, we have to deal with notes/chords merging and local rearrangements. For example, composers may choose to swap a short sequence of the notes composing a theme. Fig. 3 shows three excerpts of a musical piece from Beethoven. Each excerpt corresponds to the main motif. The first and the second excerpts contain swapped notes of the main motif. For application such as song structure discovery problem or automatic music analysis, parts of a music (verse or chorus for example) may be repeated with permuted notes.

3. POLYPHONIC ALIGNMENT

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

3.1. 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 to another one leads to the problem of computing the best permutation between both chords. Fig. 4 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.

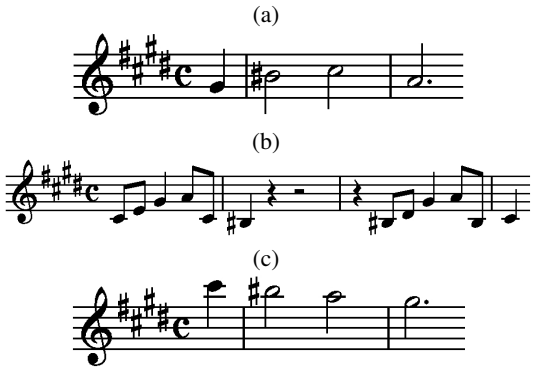

Figure 3: Similarity although permutations (a) Main motif of the 14th quatuor for piano in C# minor opus 131 from Beethoven. The motif is composed by 4 notes (sequence (1 2 3 4)). (b) First theme of the 7th movement. The 4 last notes of the two groups are permuted notes of the main motif, sequence (1 4 3 2) and (1 4 2 3) (c) Second theme of the 7th movement. The 4 notes are again a permutation of the main motif, sequence (3 2 4 1).

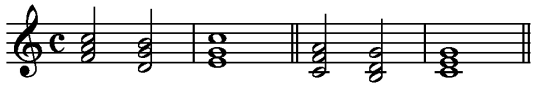

Figure 4: Similarity between inverted chords. These successive two imperfect authentic cadences in C major are similar despite the different order in the chords composing the cadences.

Considering no order and finding the best permutation allows the estimation of a high similarity between these two sequences of chords.

This optimization problem is actually a maximum score maximum bipartite matching problem and can be modeled as a weighted maximum matching algorithm [10]. Given C_1 and C_2 two chords of size n and m (we suppose $n \geq m$), we construct a graph $G(v, w) = (V, E)$ as Fig. 5:

1. *vertex set* : $V = \{s, t, e\} \cup \{s_1^1, s_1^2, \dots s_1^n\} \cup \{s_2^1, s_2^2, \dots s_2^m\}$, 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 represents ϵ;
2. *edge set* : (s, s_1^k), (e, t), (s_2^l, t) with a score 0, (s_1^k, s_2^l) with score $\sigma(s_1^k, s_2^l)$, and (s_1^k, e) with score $\sigma(s_1^k, \epsilon)$. All the edges have capacity 1 except (e, t) which capacity is $n - m$.

G is then a graph whose edges are labeled with integer capacities, non-negative costs in $\mathbb{R}$, and the maximum flow $f^* = n + m$. The complexity of computing local cost is due to this maximum cost maximum flow computation. This problem can be solved by the Edmonds and Karp's [10] algorithm improved by Tarjan [11] whose complexity is $O(|E||f^*| \log_2(|V|))$. For our graph, the maximum flow is $f^* = n + m$, the edge number is $|E| = n \times m + 2n + 2m + 3$ and the vertex number 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. We denote by σ_{bpg} the score between two chords.

In conclusion, computing alignment between 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.

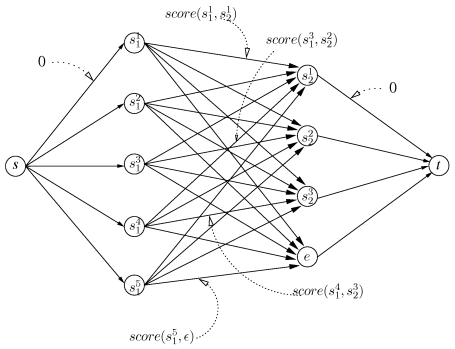

Figure 5: Resolution of the optimal permutation as a maximum cost flow problem.

3.2. Extending Mongeau-Sankoff Operations

As we already stated, an accurate algorithm for music alignment must take into account local rearrangements. Actually, these rearrangements (see Fig. 3) can be more general than the ones treated by Mongeau and Sankoff [1]. Indeed, notes can be permutated (Fig. 3c) or simultaneously permutated and merged (Fig. 3b) although the sequences of notes are still similar. We thus propose to allow a new operation that merges sub-sequences of notes in both sequences q and t simultaneously.

Music alignment between two sequences is then extended as follow:

Definition 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 pair of sub-sequence on t and q, that is $X^a = \{((i_s, i_e), (j_s, j_e))\}$ and respects the following for all $((i_s, i_e), (j_s, j_e)) \in X^a$:*

- $0 \leq i_s < i_e \leq |t|$ *and* $0 \leq j_s < j_e \leq |q|$
- $\forall((i'_s, i'_e), (j'_s, j'_e)) \in X^a$*:*
  - $i'_s \neq i_s$,
  - *if* $i'_s < i_s$ *then* $i_e \leq i_s$ *and* $j'_e \leq j_s$,
  - *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

$$G_t = \{0, \ldots, |t| - 1\} \setminus \bigcup_{((i_s,i_e),(j_s,j_e)) \in X^a} \{i_s \ldots i_e - 1\}$$

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

$$\begin{aligned} S(X^a) &= \sum_{((i_s,i_e),(j_s,j_e)) \in X^a} score(t_{i_s \ldots i_e}, q_{j_s \ldots j_e}) \\ &+ \sum_{i \in G_t} \sigma(t_i, \epsilon) + \sum_{j \in G_q} \sigma(\epsilon, q_j) \end{aligned} \quad (1)$$

The adaptation of the alignment algorithm is straightforward. In the dynamic programming matrix, it consists in computing each position $M[i][j]$ from any position $M[k][l]$ for $0 \leq k < i$, $0 \leq l < j$.

In order to compute the $score(t_{i-k \ldots i}, q_{j-l \ldots j})$ using a monophonic scoring scheme. We propose to use the monophonic scoring approach introduced in section 3.1 to compare chords. However in this case we are dealing with sequences of chords instead of single chords. In the following, we present two different methods to encode a sequence of chords into a single one.

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

Figure 6: Example of merged notes that must be considered for one step (X) of the alignment algorithm.

General scoring scheme: Let consider the sequence of chords $t_{i_s \ldots i_e}$, we define the set P of all different pitches represented in this sequence. For each pitch p in P, we associate a duration corresponding to the time elapsed from the beginning of the first occurrence of p in $t_{i_s \ldots i_e}$ to the end of the last occurrence of p in $t_{i_s \ldots i_e}$. For instance, let us consider the sub-sequence$(\{G4, B4\}, \{D2\}, \{A2\}, \{G8, B8, D8\})$. The associated set P is $\{A, B, D, G\}$. The duration associated to the pitch D is 12 which is indeed the time elapsed from the beginning of $D2$ to the end of $D8$. Finally, the chord built for this sub-sequence is $\{A2, B16, D12, G16\}$.

Once the chords corresponding to $t_{i_s \ldots i_e}$ and $q_{j_s \ldots j_e}$ are built, the score is computed using σ_{bpg} (see section 3.1). Note that if the user defines a penalty in the classical Mongeau Sankoff's operation, this penalty is added to the final score in the same way.

Using the algorithm presented in 3.1, the time required to compute the score of the case $M[i][j]$ is $O(i \times j \times C^3 \log_2(C))$ where C is maximum number of different pitches in the considered sub-sequences. This lead to an overall complexity of $O(|t|^2 \times |q|^2 \times C^3 \times \log_2(C))$.

Pitch/duration scoring scheme: Let consider the following scoring scheme between notes:

$$\begin{aligned} \sigma(n, m) &= \alpha \times \sigma_p(\text{pitch}(n), \text{pitch}(m)) \\ &+ \beta \times \sigma_d(\text{duration}(n), \text{duration}(m)) \end{aligned} \quad (2)$$

Where, $score_p$ and $score_d$ are score functions respectively between pitches and durations, and α and β two constants. In that case, the chord associated to a sub-sequence is P, the set of the different pitches that occurs in this sub-sequence. Hence the chords are composed only by pitches, we use σ_p to weight the edges of the bipartite graph to compare such chords. Finally, the score between two sub-sequences is given by:

$$score(t_{i-k \ldots i}, q_{j-l \ldots j}) = \alpha \times \sigma_{bpg}(P, Q) + \beta \times \sigma_d(K, L)$$

where P (resp. Q) is the chord associated to $t_{i-k \ldots i}$ (resp. $q_{j-l \ldots j}$) and K (resp. L) is the time elapsed from beginning of t_{i-k} (resp. q_{j-l}) to the end of t_{i-1} (resp. q_{j-1}). For the merge operation, the penalty is used in the same way.

Now let us consider the computation of $M[i][j]$. For example, let us consider the computation of the position X represented in Fig. 6. This score is obtained either:

- from f which implies the computation of scores $\sigma_{bpg}(\{A\}, \{A, D, G\})$ and $\sigma_d(2, 2)$ or,
- from e which implies the computation of $\sigma_{bpg}(\{A\}, \{A, B, D, G\})$ and $\sigma_d(2, 3)$ or,
- from d which implies the computation of $\sigma_{bpg}(\{A, D\}, \{A, D, G\})$ and $\sigma_d(4, 2)$ or,
- from c which implies the computation of $\sigma_{bpg}(\{A, D\}, \{A, B, D, G\})$ and $\sigma_d(4, 3)$ or,

- from b which implies the computation of $\sigma_{bpg}(\{A, B, D, G\}, \{A, D, G\})$ and $\sigma_d(8, 2)$ or,
- from a which implies the computation of $\sigma_{bpg}(\{A, B, D, G\}, \{A, B, D, G\})$ and $\sigma_d(8, 3)$.

One can observe that from one computation of σ_{bpg} to another one, we just add vertices in the bipartite graph. So, it is not necessary to recompute σ_{bpg} from scratch. Toroslu and Üçoluk [12] give 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 $t_{0\ldots i-1}$ and $q_{0\ldots j-1}$. 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.

4. FIRST EXPERIMENTS AND CONCLUSION

During MIREX 2006 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. Two collections were considered, and 11 queries (hummed or whistled) were proposed. The *mixed* collection is composed of 10000 randomly picked MIDI files. The *karaoke* collection is composed of about 1000 `.kar` files (Karaoke MIDI files) with mostly Western popular music. Tab. 1 presents the results obtained with these two collections and analyzed using our general polyphonic alignment algorithm and a classical alignment proposed by Uitdenbogerd [13].

Results presented in Tab. 1 clearly show that the algorithm considering new edit operations improves retrieval systems. 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 significant) is also observed for the *mixed* collection. The average precision is 0.67 instead of 0.52.

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, in particular with polyphonic queries.

In this paper, we proposed a general alignment algorithm for polyphonic alignment. In particular, this algorithm only require scoring scheme between notes (and not chords). We also introduced an original approach to compute a score between chords. Finally, it is important to notice that in practice we can bound the number of different pitches in a chord by a constant C, typically between 4 and 6. Similarly to Mongeau Sankoff [1], it is reasonable to limit the number of different pitches merged to the same value C. Another possibility is to limit the number of merged notes by another constant C'. Under these restrictions, the algorithm presented remains quadratic in the size of the input sequences.

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

Table 1: 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).

5. ACKNOWLEDGMENT

This work is part of the SIMBALS (JC07-188930) and BRASERO (ANR-06-BLAN-0045) projects, funded by the French National Research Agency (ANR).

6. REFERENCES

[1] Marcel Mongeau and David Sankoff, "Comparison of Musical Sequences," *Computers and the Humanities*, vol. 24, no. 3, pp. 161–175, 1990.

[2] Kjell Lemström and Anna Pienimäki, "Approaches for Content-Based Retrieval of Symbolically Encoded Polyphonic Music," in *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC)*, Bologna, Italy, 2006, pp. 1028–1035.

[3] Roger B. Dannenberg, William P. Birmingham, Bryan Pardo, Ning Hu, Colin Meek, and George Tzanetakis, "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)*, vol. 58, no. 5, pp. 687–701, 2007.

[4] Rainer Typke, Remco C. Veltkamp, and Frans Wiering, "Searching Notated Polyphonic Music Using Transportation Distances," in *Proceedings of the 12th ACM Multimedia Conference (MM)*, New-York, USA, 2004, pp. 128–135.

[5] T.F. Smith and M.S. Waterman, "Identification of Common Molecular Subsequences," *Journal of Molecular Biology*, vol. 147, pp. 195–197, 1981.

[6] Alexandra L. Uitdenbogerd, *Music Information Retrieval Technology*, Ph.D. thesis, RMIT University, Melbourne, Australia, July 2002.

[7] Pierre Hanna and Pascal Ferraro, "Polyphonic Music Retrieval by Local Edition of Quotiented Sequences," in *Proceedings of the 5th International Workshop on Content-Based Multimedia Indexing (CBMI)*, Bordeaux, France, 2007, pp. 61–68.

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

[9] Julien Allali, Pascal Ferraro, Pierre Hanna, and Costas Iliopoulos, "Local transpositions in alignment of polyphonic musical sequences," in *14th String Processing and Information Retrieval Symposium*, Nivio Ziviani and Ricardo Baeza-Yates, Eds. oct. 2007, vol. 4726 of *Lecture Notes in Computer Science*, pp. 26–38, Springer.

[10] Jack Edmonds and Richard M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *Journal of the Association for Computing Machinery*, vol. 19, pp. 248–264, 1972.

[11] Robert Endre Tarjan, *Data Structures and Network Algorithms*, CBMS-NFS - Regional Conference Series In Applied Mathematics, 1983.

[12] Ismail H. Toroslu and Göktürk Üçoluk, "Incremental assignment problem," *Information Sciences*, vol. 177, no. 6, pp. 1523–1529, 2007.

[13] A. Uitdenbogerd, "Variations on local alignment for specific query types," in *2nd Music Information Retrieval Evaluation eXchange (MIREX06)*, Victoria, 2006.

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