What does the Mongeau-Sankoff algorithm compute?

Henry Boisgibault, Mathieu Giraud, Florent Jacquemard

What does the Mongeau-Sankoff algorithm compute ?

Henry Boisgibault[a] Mathieu Giraud[b] Florent Jacquemard[c]

[a] Suplec, h.Boisgibault@gmail.com;
[b] CNRS, Universit de Lille, CRIStAL UMR 9189, mathieu@algomus.fr;
[c] Inria Paris, florent.jacquemard@inria.fr

ARTICLE HISTORY
October 2019

ABSTRACT
How similar are two melodies? Proposed in 1990, the Mongeau-Sankoff algorithm computes the best *alignment* between two melodies with insertion, deletion, substitution, fragmentation, and consolidation operations. This popular algorithm is sometimes misunderstood. Indeed, computing the best *edit distance*, which is the best chain of operations, is a more elaborated problem.
Our objective is to clarify the usage of the Mongeau-Sankoff algorithm. In particular, we observe that an alignment is a restricted case of edition. This is especially the case when some edit operations *overlap*, *e.g.* when one further changes one or several notes resulting of a fragmentation or a consolidation. We propose recommendations for people wanting to use or extend this algorithm, and discuss the design of combined or extended operations, with specific costs.

KEYWORDS
Music similarity; edit distance; alignment; dynamic programming; consolidation; fragmentation

1. Introduction

Assessing music similarity is a fundamental task in Music Information Retrieval (MIR) studies. Mongeau and Sankoff (1990) proposed to extend sequence comparison algorithms computing an *edit distance* (the minimal cost for transforming one given string into one other, using a definite set of weighted primitive operations) in order to compare melodies. At that time, Marcel Mongeau was a bachelor student in computer science, and also played and produced experimental music with friends. David Sankoff was already an expert in string comparison, notably for applications in bioinformatics, and worked in the *Centre de recherches mathématiques* (CRM) at *Université de Montréal*. David Sankoff taught a lecture on sequence comparison and bioinformatics and proposed this topic to Marcel Mongeau[1]. Both authors are now successful scholars, but neither of them continued research in MIR – Marcel Mongeau works on optimization and operations research, and David Sankoff on bioinformatics. The algorithm of Mongeau and Sankoff (1990) became very popular, and this paper is one of the most cited in the MIR field, being referenced by more than 300 academic works (Figure 1).

[1] M. Mongeau, personal communication

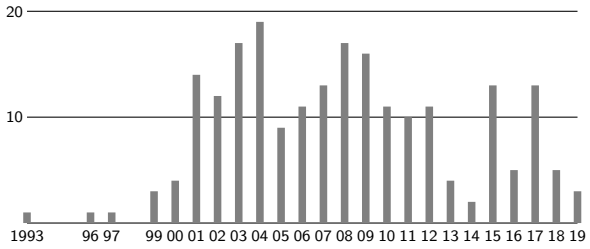

Figure 1. Citations of the Mongeau-Sankoff algorithm throughout the years, data from semanticscholar.

Today, there is a lot of research in music similarity that spans other music concepts than melodies, using possibly harmony or structure (Berit, de Haas, Volk, and van Kranenburg, 2013; Janssen, van Kranenburg, and Volk, 2015). Even when one restricts to melodies, some methods do not rely anymore on string comparison, like geometric matching (Ukkonen, Lemström, and Mäkinen, 2003), or, more recently, fuzzy approaches, wavelets or deep learning (Cheng, Fukayama, and Goto, 2018; Velarde, Weyde, and Meredith, 2013). However, the classical Mongeau-Sankoff algorithm is still used and extended in many studies, even if most of the people do not use the exact *cost functions* proposed by the authors.

One of the core ideas of the authors is to use, together with the standard primitive operations of the Levenshtein distance (namely single-character *insertion*, *deletion* and *substitution*) the new operations of *consolidation* (a few characters being compressed into a single one) and *fragmentation* (a single character being expanded into a few), for a better match with musical content. Consolidation and fragmentation operations are similar to the *compression* and *expansion* operations of Dynamic Time-Warping algorithms used in music but also for *e.g.* speech recognition, where they allow to compare two speech recordings with different variations in speed (Kruskal and Liberman, 1999).

The Mongeau-Sankoff algorithm is perceived as relevant to measure music similarity, and was evaluated against other algorithms (Gomez, Abad-Mota, and Ruckhaus, 2007; Grachten, Arcos, and de Mántaras, 2002). Indeed, fragmentations and consolidations are frequently used in variations (Figure 5 and 6, later in the article), but also in songs and vocal music, where several lyrics are used on a same melodic contour (Figure 2). Several authors used or extended these consolidation/fragmentation operations, and sometimes proposed other edit operations, for a wide range of MIR applications, such as query-by-humming (Dannenberg, Birmingham, Pardo, Hu, and Meek, 2006; Dannenberg and Hu, 2004; Hu and Dannenberg, 2002), musical dictations marking (Tremblay and Champagne, 2002), ornementation (Grachten, Arcos, and de Mántaras, 2005), performers behavior annotation (Arcos, Grachten, and de Mántaras, 2003), polyphonic alignment (Hanna, Robine, Ferraro, and Allali, 2008), rhythm alignment (Levé, Groult, Arnaud, Séguin, Gaymay, and Giraud, 2011), or variation analysis and sampling (Giraud, Déguernel, and Cambouropoulos, 2013; Pachet, Papadopoulos, and Roy, 2017).

The Mongeau-Sankoff algorithm is presented as a set of Dynamic Programming equations. The common idea is that it compares two melodies while finding *the best combination of substitutions, deletions, insertions, consolidations, and fragmentations.* Indeed, it computes the best *alignment* (called *trace* by Mongeau and Sankoff, see Figure 3) between two melodies with these operations, that is a correspondence between their respective positions describing applications of primitive operations:

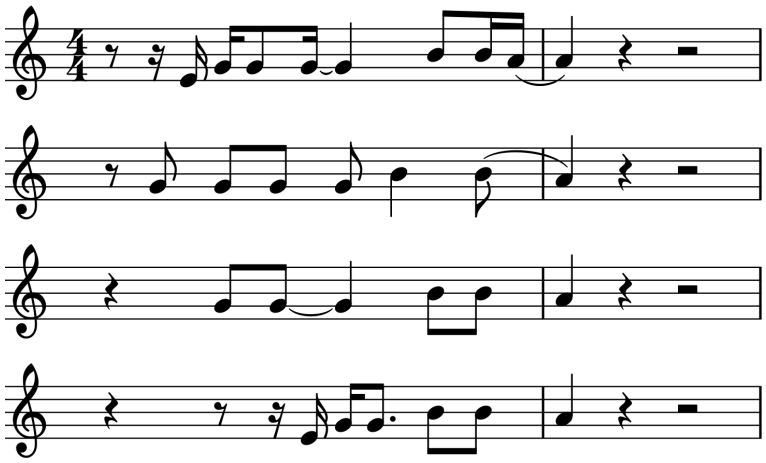

Figure 2. Patterns in the verse of *Imagine*, by John Lennon and Yoko Ono, can be seen as fragmentations of a melody G/B/A. The same melody is found in the piano theme, not shown here.

> *"It can be shown that the optimal trace linking two musical sequences is obtained through the use of pointers [in the] recurrence equation and the best alignment is found by following the pointers starting at [the last case of the recurrence table]"*
>
> *(Mongeau and Sankoff, 1990)*

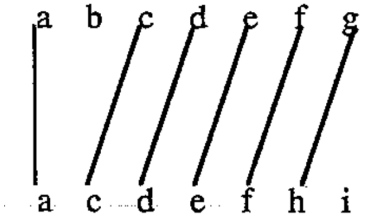

Figure 3. Trace linking two sequences, from (Mongeau and Sankoff, 1990).

Aim and Contents

We aim here to lay out precisely what Dynamic Programming algorithms like the Mongeau-Sankoff algorithm does compute, stressing this difference between alignments and edit distances. Note that some relations between alignments and edit distances, not applied to music, were already studied by Ukkonen (1985). In particular, we observe that an edit distance with fragmentations and consolidations is, in general, not even computable, even if many practical cases are tractable.

After some definitions in Sections 2 (operations), 3 (edit distances) and 4 (alignments), we show in Section 5 that it is sometimes different to compute the *best alignment* and the *edit distance* between two melodies, as some transformations by edit operations cannot be viewed as an alignment. In other words, an alignment is a restricted case of edition. This is especially the case when some edit operations *overlap*,

operations	cost	
s	1	Hamming distance
sdi	1	Levenshtein distance
sdiCF	1	
some in sdicf family	1, adding w_{pitch} or/and w_{dur} for non-strict f/c	

Table 1. Some rulesets and costs used in this paper. The actual (Mongeau and Sankoff, 1990) algorithm used more elaborated costs.

e.g. when one further changes one or several notes resulting of a fragmentation or a consolidation.

This case was partly taken into account by Mongeau and Sankoff, who encoded into the costs of fragmentation or consolidation other potential operations. Nevertheless, some musically relevant cases are not optimally handled by the original algorithm, such as the one depicted on Figure 4, that involves a fragmentation followed by a consolidation. Section 6 discusses the design of such combined or extended operations, with specific costs, and opens perspectives on future research.

This paper is intended for people working in computer music wanting to understand, use, or extend a Mongeau-Sankoff-like algorithm for melodic similarity computation. We present in (Giraud and Jacquemard, 2019) a more theoretical content on the computability of such edit distances.

2. Operations, rulesets and costs

We consider a fixed finite set $\mathcal{N}$ of symbols representing *notes* and denoted by α, β... Each symbol α gathers two attributes of pitch and duration, respectively denoted by $\mathsf{pitch}(\alpha)$ and $\mathsf{dur}(\alpha)$. A monophonic sequence of notes, or, for short, a *melody*, is a sequence over $\mathcal{N}$. The set of sequences over $\mathcal{N}$ is denoted $\mathcal{N}^*$ and ε denotes the empty sequence.

Operations and rulesets

An *edit operations* is a pair $(\ell \to r)$ with $\ell, r \in \mathcal{N}^*$. A *ruleset* E is a set of edit operations of one of the following types, which are commonly applied to melodies:

(s) substitution: $\alpha \to \beta$,
(i) insertion: $\varepsilon \to \beta$,
(d) deletion: $\alpha \to \varepsilon$,
(f) fragmentation: $\alpha \to \alpha_1 \dots \alpha_k$, with $k \geq 0$,
(c) consolidation: $\alpha_1 \dots \alpha_k \to \alpha$, with $k \geq 0$.

Note that (s) and (d) are particular cases of (f), with respectively $k = 1$ and $k = 0$, and (s) and (i) are particular cases of (c).

We also consider *strict fragmentation* (F) that splits one note into a set of notes of the same pitch and of the same total duration, that is, such that $\sum_{i=1}^{k} \mathsf{dur}(\alpha_i) = \mathsf{dur}(\alpha)$ and $\mathsf{pitch}(a_1) = \ldots = \mathsf{pitch}(\alpha_k) = \mathsf{pitch}(\alpha)$, and *strict consolidation* (C) that proceeds in the other direction, with the same restrictions.

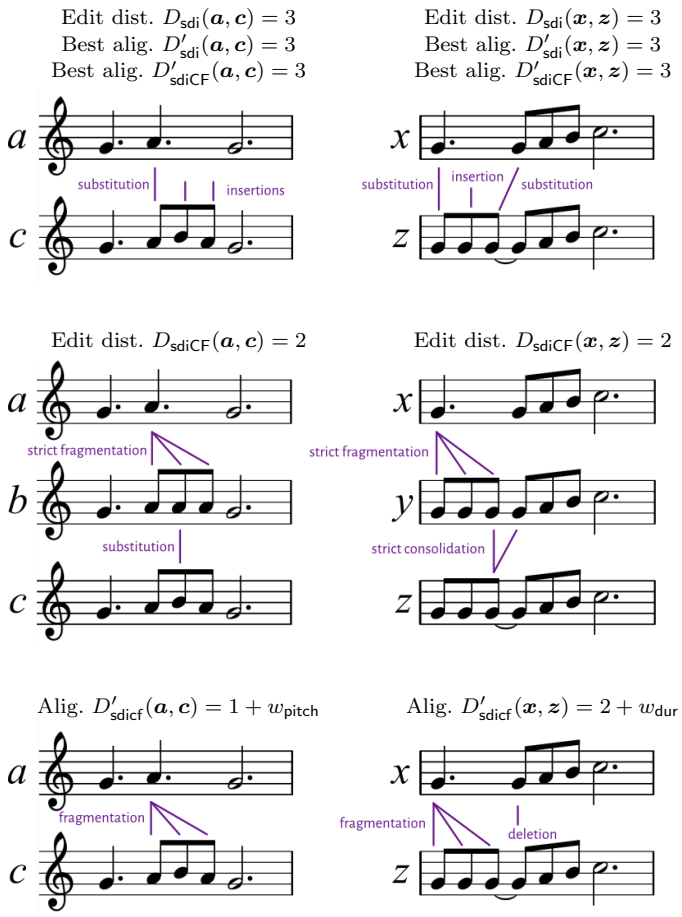

Figure 4. Two independent examples (left and right columns) where best alignments do not correspond to the edit distance for some edit operations and costs. Rulesets and costs are defined on Table 1.
(Top) With insertion/deletion/substitutions, best alignements between melodies $\boldsymbol{a}$ and $\boldsymbol{c}$ (left) and between melodies $\boldsymbol{x}$ and $\boldsymbol{z}$ (right) are the edit distances.
(Middle) With strict fragmentation and consolidations with a cost of 1, some optimal sequence of edit operations are not alignments: They are overlapping, here with a substitution in the middle of a fragmentation (left) and a consolidation after a fragmentation (right). Alignments can here be represented by using other melodies $\boldsymbol{b}$ and $\boldsymbol{y}$.
(Bottom) When fragmentation and consolidation can include pitch or total duration change, some alignments can be closer or equal the edit distance, depending on the costs w_{pitch} and w_{dur} for these operations.

Chain of positioned operations

A *positioned operation* of a ruleset E is a pair $\pi = \langle \ell \to r, i \rangle$ made of an edit operation of E and a natural number, meaning that $\ell \to r$ is applied at position i. For two melodies s, t of $\mathcal{N}^*$, we write $s \xrightarrow{\pi} t$ if for some $u, v \in \mathcal{N}^*$, $s = u\,\ell v$, $t = urv$, and $i = |u|$. For a *chain* of positioned operations $\vec{\pi} = \pi_1 \pi_2 \dots \pi_n$, with $\pi_j = \langle \ell_j \to r_j, i_j \rangle$ for all $j = 1..n$, we write $s \xrightarrow{\vec{\pi}} t$ if $s = s_0 \xrightarrow{\pi_1} s_1 \dots \xrightarrow{\pi_n} s_n = t$ for some $s_0, \dots, s_n \in \mathcal{N}^*$.

For example, we consider the three melodies of the left of Figure 4: $\boldsymbol{a} = a_1\,a_2\,a_3$, $\boldsymbol{b} = b_1\,b_2\,b_3\,b_4\,b_5$, and $\boldsymbol{c} = c_1\,c_2\,c_3\,c_4\,c_5$. Going from $\boldsymbol{a}$ to $\boldsymbol{b}$ is done by the fragmentation of the dotted quarter A into three eights, $a_2 \xrightarrow{frag} b_2\,b_3\,b_4$ (and $a_1 = b_1$, $a_3 = b_5$), and going from $\boldsymbol{b}$ to $\boldsymbol{c}$ is done by the (pitch) substitution of one eight $b_3 \xrightarrow{subst} c_3$ (and

$c_1 = b_1$, $c_2 = b_2$, $c_4 = b_4$, $c_5 = b_5$). It is written as positioned operations $\boldsymbol{a} \xrightarrow{\langle \mathsf{frag},1\rangle} \boldsymbol{b}$ and $\boldsymbol{b} \xrightarrow{\langle \mathsf{subst},2\rangle} \boldsymbol{c}$, and, as a chain, $\boldsymbol{a} \xrightarrow{\vec{\pi}=\langle \mathsf{frag},1\rangle\langle \mathsf{subst},2\rangle} \boldsymbol{c}$.

Costs

In order to look to the best way to transform a melody into another one, one has to be able to measure the likelihood of a chain of operations. Each operation $\ell \to r$ of a ruleset E is associated a strictly positive cost $\delta(\ell \to r)$. The cost of a chain of positioned operations is then the sum of the individual costs: $\delta(\vec{\pi}) = \sum_{i=1}^{n} \delta(\ell_i \to r_i)$.

Ideally, cost precisely model the likelihood of an operation. In bioinformatics, to compare proteins seen as sequences over the 20-letter alphabet of amino acids, the BLOSUM62 substitution matrix (Henikoff and Henikoff, 1992), learned on aligned data, reflects the logarithm of the probability of substitution of any two amino acids, and the cost of a chain estimates the probability of this chain of operations.

Designing actual cost functions play an important role in achieving musically relevant results and will be discussed in Section 6. To keep the argument simple, we will use here very simple costs: 1 for every (sdiCF) operation, and additional costs when one allows non-strict fragmentations and consolidations. Table 1 shows these rulesets and costs.

3. Edit distance: chains of operations

Edit distance quantifies the proximity of two sequences by asserting the *minimum cost of operations* transforming one into the other one. The idea here is that when operations are "atomic", one can combine them. Formally, the edit distance between s and t *wrt* a ruleset E is:

$$D_E(s,t) \;=\; \min\bigl\{\delta(\vec{\pi}) \;\big|\; s \xrightarrow{\vec{\pi}} t\bigr\}.$$

This implies that, for two rulesets E and F with $E \subset F$, we have

$$D_E(s,t) \geq D_F(s,t) \tag{1}$$

Indeed, all chain of operations using E are chains of operations using F, and thus having a larger ruleset may lower the minimum cost.

For example, taking all substitution, deletion and insertion operations (E_{sdi}) with a cost of 1, for the melodies in Figure 4 (left), we have $D_{\mathsf{sdi}}(\boldsymbol{a},\boldsymbol{c}) = 3$, with a chain of one (duration) substitution and insertions of two eights (we write D_{sdi} instead of $D_{E_{\mathsf{sdi}}}$ to enlighten notations).

If we add the strict fragmentations and consolidations, also with cost 1, the edit distance is now $D_{\mathsf{sdiCF}}(\boldsymbol{a},\boldsymbol{c}) = 2$ with the chain presented above with one fragmentation and one substitution. Here $D_{\mathsf{sdiCF}} \leq D_{\mathsf{sdi}}$ because $E_{\mathsf{sdi}} \subset E_{\mathsf{sdiCF}}$.

Note that as soon as a ruleset contains at least E_{di}, that is all deletion and insertion operations, there is always at least one way to transform one melody s into another one t, by deleting all notes of s then inserting all notes of t. However, in the general case, with arbitrary rulesets, there can be no chain of operations of E transforming s into t, and in this case we set $D_E(s,t) = +\infty$.

Computability of the edit distance

The edit distance D_E can be efficiently computed in some common cases (see Section 5). On may even observe however that, in general, the problem whether $D_E(s,t) < +\infty$, given some arbitrary ruleset E and sequences s and t, is equivalent to the problem of the existence of a chain $\vec{\pi}$ of E such that $s \xrightarrow{\vec{\pi}} t$ (a problem called *reachability* of t from s using E).

It is possible to encode the computations of a given Turing machine $\mathcal{M}$ in these settings. More precisely, every configuration of $\mathcal{M}$ can be encoded into a finite sequence of $\mathcal{N}^*$, and every computation step $\mathcal{M}$ can be simulated by a combination of operations of types (c), (f) (with at most 4 symbols), and (i), (d). Hence, there exists a finite ruleset $E(\mathcal{M})$ such that, $\mathcal{M}$, when starting in a configuration encoded into $s \in \mathcal{N}^*$, can terminate its computation into another configuration encoded into $t \in \mathcal{N}^*$, if and only if $D_{E(\mathcal{M})}(s,t) < +\infty$. As the former problem is known to be undecidable, there is no hope to be able to compute $D_E(s,t)$ in general. This reachability can also be encoded into another undecidable problem (Giraud and Jacquemard, 2019).

4. Alignments: chains of *non-overlapping* operations

Alignments between sequences represent *some* chains of operations by layering down how some elements of the first sequence are transformed into elements of the second one. To conveniently represent the alignment on two lines, a key principle is that the operations are non-overlapping: Blocks of elements from the first sequences are transformed into blocks of elements of the second sequence.

Alignment

An *alignment* (that was called a *trace* by Mongeau and Sankoff) $\vec{\pi} = \pi_1\pi_2\dots\pi_n$ is a chain of consecutive *non-overlapping* positioned operations of E, that is, such that $i_j \geq i_{j-1} + |r_{j-1}|$ for all $j = 2..n$, writing $\pi_j = \langle \ell_j \to r_j, i_j \rangle$. We call $D'_E(s,t)$ the optimal cost of an alignment between s and t using E:

$$D'_E(s,t) = \min\left\{\delta(\vec{\pi}) \;\middle|\; s \xrightarrow{\vec{\pi}} t, \vec{\pi} \text{ is a alignment}\right\}$$

For two rulesets E and F with $E \subset F$, all alignements using E are alignements using F, thus we have again, as in (1)

$$D'_E(s,t) \geq D'_F(s,t) \tag{2}$$

Figure 4 (top, left) depicts an example of alignment $\pi_1\pi_2\pi_3$ between the melodies $\boldsymbol{a}$ and $\boldsymbol{c}$, where π_1 is a substitution (preserving pitch but reducing duration) applied at position 1, and π_2, π_3 are two insertions applied respectively at positions 2 and 3. This alignment is optimal for E_{sdi} and a, c, hence we have $D'_{\mathsf{sdi}}(\boldsymbol{a},\boldsymbol{c}) = 3$. It also holds that $D_{\mathsf{sdi}}(\boldsymbol{a},\boldsymbol{c}) = 3$: the best alignment is here the edit distance.

Computing best alignments with Dynamic Programming

Let $S(i,j)$ be the cost of the best alignment $D'_E(\alpha_1\dots\alpha_i, \beta_1\dots\beta_j)$. As the alignments are not overlapping, Bellman's principle of optimality can be applied. When the ruleset

is finite, this cost can be computed by Dynamic Programming, as shown in 1985 by Ukkonen (Algorithm 13 of (Ukkonen, 1985)):

$$S(i,j) = \min \Big\{ \; S(i-k, j-\ell) + \; \delta(\alpha_{i-k+1} \dots \alpha_i \to \beta_{j-\ell+1} \dots \beta_j) \tag{3}$$

If the ruleset is E_{sdicf}, this is the well-known Mongeau and Sankoff (1990) algorithm:

$$S(i,j) = \min \begin{cases} S(i-1,j-1) \text{ if } \alpha_i = \beta_j & \text{(match)} \\ S(i-1,j-1) + \delta(\alpha_i \to \beta_j) & \text{(substitution)} \\ S(i-1,j) + \delta(\alpha_i \to \varepsilon) & \text{(deletion)} \\ S(i,j-1) + \delta(\varepsilon \to \beta_j) & \text{(insertion)} \\ S(i-k,j-1) + \delta(\alpha_{i-k+1} \dots \alpha_i \to \alpha_j) & \text{(consolidation)} \\ S(i-1,j-k) + \delta(\alpha_i \to \beta_{b-k+1} \dots \beta_j) & \text{(fragmentation)} \end{cases}$$

To compute the best *local* alignment $D'_E(\alpha_{i'} \dots \alpha_i, \beta_{j'} \dots \beta_j)$, one may use the same equations adding a 0 in each minimum. In both cases, computing $S(i,j)$ is done in time $O(i \cdot j \cdot |E|)$.

5. Do best alignments correspond to the edit distance ?

As every alignment is a sequence of positioned operations, we always have

$$D'_E(s,t) \geq D_E(s,t) \tag{4}$$

This inequality can be strict in some cases, as a sequence of positioned operations may not be an alignment.

With insertions, deletions and substitutions, best alignments correspond to the edit distance

In the case $E = E_{\mathsf{sdi}}$, we have $D_{\mathsf{sdi}} = D'_{\mathsf{sdi}}$. Indeed, there is always an optimal sequence of *non-overlapping* operations yielding to the (minimal) edit distance.

More formally, when the costs respect the triangle inequality, for every chain of positioned operations $\vec{\pi}$ such that $s \xrightarrow{\vec{\pi}} t$, there exists an alignment $\vec{\tau}$ such that $s \xrightarrow{\vec{\tau}} t$, with $\delta(\vec{\tau}) \leq \delta(\vec{\pi})$. Thus D_{sdi} can be computed using the above Dynamic Programming equation (3).

This is true for the Levenshtein distance (E_{sdi}, all costs 1), but also for any other costs, leading to the Needleman and Wunsch (1970) and Smith and Waterman (1981) algorithms used in bioinformatics sequence comparison. The popular conception such that "the best alignment is the edit distance" is thus true on these common cases.

In general, best alignments do not always correspond to edit distance

As we have seen that the edit distance D_E is *not computable* in general (Giraud and Jacquemard, 2019), whereas the best alignment D'_E is always computable in polynomial time. Hence it comes with no surprise that the two concepts deviate at some point. Indeed, for some rulesets more elaborated than D_{sdi}, it might hold that $D'_E(s,t) > D_E(s,t)$, in the case where the optimal sequence of positioned operations involves *overlapping positioned operations* (Ukkonen, 1985).

In Figure 4 (both left and right), adding strict fragmentations and consolidations to the set of operations does not change the optimal alignment, with $D'_{\mathsf{sdi}}(\boldsymbol{a},\boldsymbol{c}) = D'_{\mathsf{sdiCF}}(\boldsymbol{a},\boldsymbol{c}) = 3$ and $D'_{\mathsf{sdi}}(\boldsymbol{x},\boldsymbol{z}) = D'_{\mathsf{sdiCF}}(\boldsymbol{x},\boldsymbol{z}) = 3$ (top), whereas the edit distances with the same operations are 2 (middle). The reason is that in both cases, the last operation of the optimal chain of edit operations overlaps with the result of a previous fragmentation: on the left, the optimal chain is $\boldsymbol{a} \xrightarrow{\vec{\pi}=\langle \mathsf{frag},1\rangle\langle \mathsf{subst},2\rangle} \boldsymbol{c}$ and the second position $2 < 1+3$ (1 is the position of fragmentation $a_2 \xrightarrow{\mathsf{frag}} b_2\, b_3\, b_4$ and 3 is its length); on the right, the optimal chain is $\boldsymbol{a} \xrightarrow{\vec{\pi}=\langle \mathsf{frag}',0\rangle\langle \mathsf{cons},2\rangle} \boldsymbol{c}$ and similarly, $2 < 0+3$. Therefore in both cases, the restriction that $i_j \geq i_{j-1} + |r_{j-1}|$ in the definition of alignement is not satisfied.

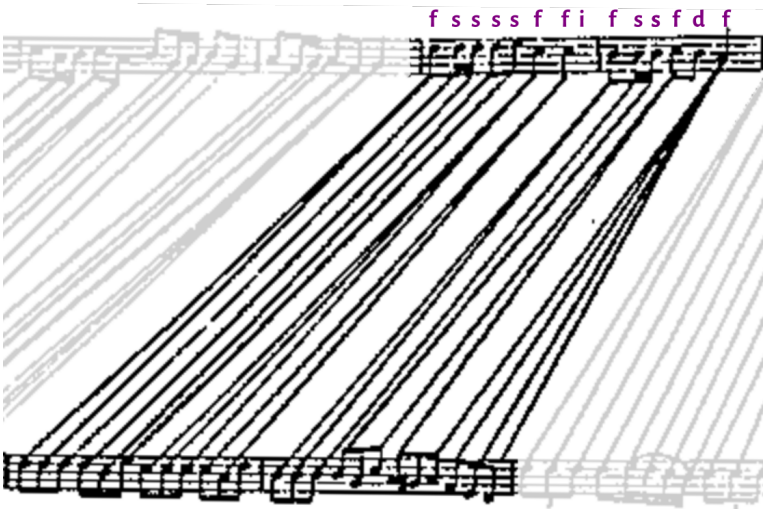

Figure 5. Alignments between variations 3 and 7 by M. Duchesnes on *Ah vous dirai-je maman*. Figure from the original article (Mongeau and Sankoff, 1990), annotations are ours. While they are both in binary meter, Variation 3 is with binary subdivisions and variation 7 with ternary subdivisions, making note-by-note alignment difficult. Here *none* of the fragmentations (f) do preserve the total duration, whereas all of the substitutions (s) change durations. This alignment involves 5 fragmentations, 6 substitutions, 1 deletion and 1 insertion, for a cost of about $14 + 12w_{\mathsf{dur}}$.

Impact on computed similarities

We take the same example that was used in 1990, on variations on *Ah vous dirai-je maman*, by Mario Duschenes (1962) in his recorder method, which was very popular in Québec and more generally in Canada. Even if these variations may have been inspired by Mozart's own variations K265/300e, they are not the ones of Mozart nor an arrangement, but an original work by Mario Duchesnes.

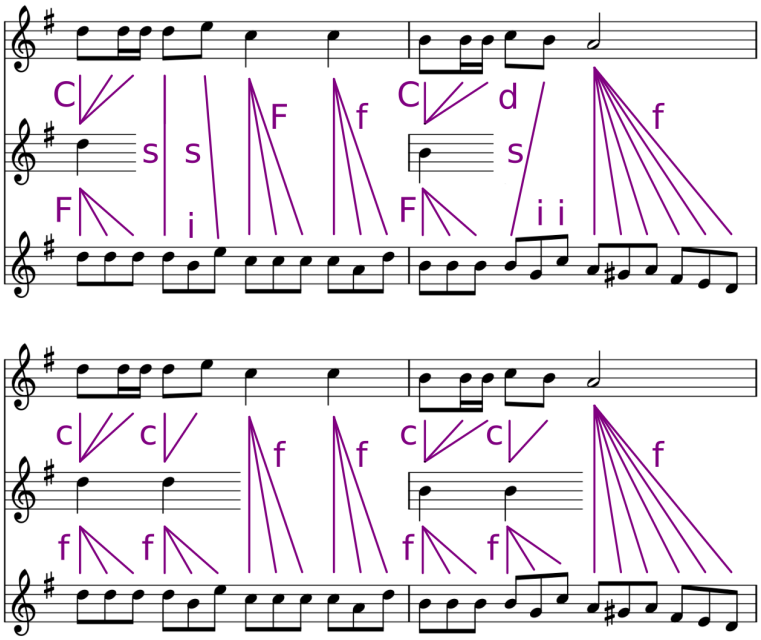

Figure 6. Alignments between variations 3 and 7 by M. Duchesnes on *Ah vous dirai-je maman*, compare to Figure 5. Chains with overlapping operations. All considered consolidations and fragmentations preserve the total duration, as Variation 7 is written with triplets.
(Top) This chain of operation involves 2 strict consolidation, 3 strict fragmentations, and 3 other fragmentations. Other places are use 6 s/d/i operations similar to the original aligment of the Figure 5. The total cost is $14 + 3w_{\mathsf{pitch}}$.
(Bottom) Here the chain contains only consolidation and fragmentation operations (11 operations, including 5 strict ones). The total cost is $11 + 6w_{\mathsf{pitch}}$.

Comparing Variations 3 and 7 confirms that a proper computation of edit distances, with fragmentations and consolidations, yields here a more musically relevant similarity than the alignments output by the Mongeau-Sankoff algorithm (Figures 5 and 6). As usual, note that the actual edit distance and optimal costs depend on the chosen costs. This example is particularly challenging as both variations do not have the same subdivision of time. An extreme way of chaining operations, consolidating every beat then refragmenting them (Figure 6, bottom), yields here a very low edit distance. This also confirms the "central role of the theme", as already noted by Mongeau and Sankoff: Somehow a very relevant way to compare variations is to align each of them to the theme.

6. Discussion and Perspectives: Combined and Extended Operations

The previous sections showed that, while proper edit distances with fragmentations and consolidation yield interesting melodic similarities, they are not as easy to compute as best alignments. One could use other algorithms – such as geometric algorithms or algorithms learning similarity from annotated data – to assess the music similarity between melodies. But edit distances are a good way to model how musical patterns may evolve. Dynamic Programming methods are relatively efficient and simple to model. As some of these cases may be musically relevant, especially in paradigmatic music analysis, can we compute alignments that reflect or approximate the edit distance?

Exact Computation of Edit Distance

Edit distance can be seen as the length or the weight of the shortest path in a *edition graph* where vertices are melodies and edges the edit operations. Assuming that this graph is finite, the Dijkstra's algorithm finds the shortest path in $O(e + v \log v)$, where e is the number of edges and v the number of vertices (Dijkstra, 1959). However, this graph is almost always infinite, for example as soon as there are insertions or fragmentations.

We have seen that for E_{sdi}, the edit distance coincides with the notion of best alignment that can be computed efficiently with (3). Besides this well-known case, further research should explore which edit distances considering *some* fragmentations and consolidations can be efficiently computed, in particular for rulesets containing only consolidations and deletions, or only fragmentations and insertions (Giraud and Jacquemard, 2019).

Over-Approximation of Edit Distance

Mongeau and Sankoff were themselves aware that the similarity they compute was not a distance, and proposed in their original article the use of elaborated *costs* within the fragmentation and consolidations. Researching on the best costs to model music similarity is still an open problem, but we restate here the link between these costs and sequence of edit operations.

To cope with the fact that edit distance can not always be exactly computed, a simple idea is to have *combined or extended operations* that will finally be sequenced in a non-overlapping way to allow a proper use of a generalized Algorithm (3). One can indeed create new edit operations that combines two or more successive positioned edit operations, possibly overlapping, in E. For example, the new operation $(\alpha \to \alpha\beta\alpha)$ combines the fragmentation $(\alpha \to \alpha\alpha\alpha)$ with a substitution $(\alpha \to \beta)$ in second position.[2] The cost of this new operation is the sum of the two former operations. This new "fragmentation-then-substitution" operation is one (non-strict) (f) operation, that allows, at once, to align melodies $\boldsymbol{a}$ with $\boldsymbol{c}$ in Figure 4 (bottom left). If one strictly combines the costs, the cost of this operation is here $1 + w_{\mathsf{pitch}} = 2$, allowing here to have a alignment cost $D'_{\mathsf{sdicf}}(\boldsymbol{a}, \boldsymbol{c})$ equal to the edit distance. Similarly, a combined operation transforming x into z (right of Figure 4) would be on the form $(\alpha\beta \to \gamma\delta)$.

We call E^2 the set obtained by completing E with all edit operations combining two operations in E. We can observe on the one hand that $D_{E^2}(s,t) = D_E(s,t)$ for all melodies s, t: Applying a combined operation in a chain is just a shortcut to the application of two operations of E, with the same cost. But on the other hand, it holds by Equation (2) that $D'_{E^2}(s,t) \leq D'_E(s,t)$: Such shortcuts allow, in some cases, to turn an overlapping operation chain into an alignment, without overlaps. In other terms, D'_{E^2} is a better approximation of D_E than D'_E. The combination can be continued into sets E^3, E^4... until a *closure* $E^* = \bigcup_{i \geq 1} E^i$.

By construction, $D_E(s,t) = D'_{E^*}(s,t)$. In general, E^* will be infinite – and even with the ruleset $E = E_{\mathsf{sdi}}$, E^* is infinite. We may however consider any finite subset $E^\bullet \subset E^*$ with $E \subset E^\bullet$ to compute with Equation (3) an approximation of D_E better than D'_E.

[2] In fact, it is a combination of positioned operations: $\langle \alpha \to \alpha\alpha\alpha, i \rangle$ and $\langle \alpha \to \beta, i+1 \rangle$ gives $\langle \alpha \to \alpha\beta\alpha, i \rangle$.

Indeed, by application of Equation (2), we have:

$$D_E(s,t) = D'_{E^*}(s,t) \leq D'_{E^\bullet}(s,t) \leq D'_E(s,t).$$

However, even when $E = E_{\mathsf{sdicf}}$, the set E^2 is difficult to construct, and there are still chains of operations on E that will not be alignments on E^2, as for example two consolidations overlapping both ends of a pattern created by a fragmentation (in that case this combination will be in E^3).

Note that E is *implicitly represented* in Dynamic Programming methods. For exemple, the substitution on the form $\alpha \to \beta$ has a quadratic number of instances, substituting each note to another one, but they are handled by a single line in the Algorithm (3). The possible dependence on α and β is hidden in the cost $\delta(\alpha, \beta)$.

On the contrary, *explicitly representing* E^2 or any other $E^\bullet \subset E^*$ may lead to combinatorial explosion, beginning with operations such as $(\alpha\beta \to \gamma\delta)$ seen above. In the general case, this explosion can not be prevented as it reflects the exponential number of paths into an edition graph. Further research should be done on the computability of such combined operations in some restricted cases.

Using and designing costs on extended operations

One can nevertheless advise the proper use of consolidations and fragmentations. The original costs of the Mongeau-Sankoff algorithm involve pitches (and intervals) and durations:

$$S(i,j) = \min \begin{cases} \dots \\ S(i-k, j-1) + \delta(\alpha_{i-k+1} \dots \alpha_i \to \beta_j) \\ S(i-1, j-k) + \delta(\alpha_i \to \beta_{j-k+1} \dots \beta_j) \end{cases}$$

Researches should continue on designing cost functions δ encoding some operations of E or E^* as insertion, substitutions, or even other operations. Indeed, defining operations, costs, and thresholds is an opportunity to think on music similarity (Berit et al., 2013; Cambouropoulos, 2009; Janssen, van Kranenburg, and Volk, 2015). Costs are thus not only implementation details but fundamental to any such a method.

When comparing most Western common practice music, the impact of the operations on *tonality* and *meter* can be assessed through costs. For example, substitutions, consolidation, and fragmentation changing durations break the metric structure and may have higher costs. Substitutions of pitches may also be weighted differently, for example favoring neighbor notes or notes within the same consonant chord. Such "close" substitutions are commonly used in variations (see again Figure 6), or, more generally, in elaborations of a pattern.

As suggested by Giraud, Déguernel, and Cambouropoulos (2013), one could even use extended operations such as a *transformation operation*, editing as once several notes into several others, that is taking exactly the Ukkonen (1985) algorithm (3). This opens both musicological and computational questions on how to pertinently and efficiently evaluate $\delta(\alpha_{i-k+1} \dots \alpha_i \to \beta_{j-\ell+1} \dots \beta_j)$. The evaluation of δ could be delegated to other techniques, either computing exact edit distances (for example on an edition graph), or using other ways to model small transformations. Once δ

is defined, the algorithm (3) would again compute the best chain of *non-overlapping* extended editions.

References

Arcos, Josep Lluís, Maarten Grachten, and Ramon López de Mántaras. 2003. "Extracting Performers' Behaviors to Annotate Cases in a CBR System for Musical Tempo Transformations." In *Case-Based Reasoning Research and Development,* 20–34.

Berit, Janssen, W. Bas de Haas, Anja Volk, and Peter van Kranenburg. 2013. "Finding repeated patterns in music: State of knowledge, challenges, perspectives." In *International Symposium on Computer Music and Multidisciplinary Research (CMMR 2013),* 277–297.

Cambouropoulos, Emilios. 2009. "How similar is similar?." *Musicae Scientiae* 13 (1s): 7–24.

Cheng, Tian, Satoru Fukayama, and Masataka Goto. 2018. "Comparing RNN parameters for melodic similarity." 763–770.

Dannenberg, Roger B., William P. Birmingham, Bryan Pardo, Ning Hu, and Colin Meek. 2006. "A Comparative Evaluation of Search Techniques for Query-by-Humming Using the MUSART Testbed." *J. of the American Society for Information Science and Technology* 58 (5): 687–701.

Dannenberg, Roger B., and Ning Hu. 2004. "Understanding search performance in Query-By-Humming systems." In *International Conference on Music Information Retrieval (ISMIR 2004),* .

Dijkstra, Edsger W. 1959. "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik* 1 (1): 269–271.

Duschenes, Mario. 1962. *Variations on Twinkle, Twinkle, Little Star*, 69–72.

Giraud, Mathieu, Ken Déguernel, and Emilios Cambouropoulos. 2013. "Fragmentations with pitch, rhythm and parallelism constraints for variation matching." In *International Symposium on Computer Music and Multidisciplinary Research (CMMR 2013),* 298–312.

Giraud, Mathieu, and Florent Jacquemard. 2019. "Weighted Automata Computation of Edit Distances with Consolidations and Fragmentations." https://hal.archives-ouvertes.fr/hal-01857267.

Gomez, Carlos, Soraya Abad-Mota, and Edna Ruckhaus. 2007. "An analysis of the Mongeau-Sankoff algorithm for music information retrieval." In *International Conference on Music Information Retrieval (ISMIR 2007),* 193–199.

Grachten, Maarten, Josep-Llúis Arcos, and Ramon López de Mántaras. 2002. "A Comparison of Different Approaches to Melodic Similarity." In *Int. Conf. on Music and Artificial Intelligence (ICMAI 2002),* .

Grachten, Maarten, Josep Lluís Arcos, and Ramon López de Mántaras. 2005. "Evolutionary Optimization of Music Performance Annotation." In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2005),* 347–358.

Hanna, Pierre, Matthias Robine, Pascal Ferraro, and Julien Allali. 2008. "Improvments of Aligmnement Algorithms for polyphonic music Retrieval." In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2008),* 244–251.

Henikoff, S., and J. Henikoff. 1992. "Amino acid substitution matrices from protein blocks." *PNAS* 89: 10915–10919.

Hu, Ning, and Roger B. Dannenberg. 2002. "A Comparison of Melodic Database Retrieval Techniques Using Sung Queries." In *Int. Joint Conference on Digital Libraries (JCDL 2002),* 301–307.

Janssen, Berit, Peter van Kranenburg, and Anja Volk. 2015. "A comparison of symbolic similarity measures for finding occurrences of melodic segments." In *International Society for Music Information Retrieval Conference (ISMIR 2015),* 659–665.

Kruskal, Joseph B., and Mark Liberman. 1999. "The Symmetric Time-Warping Problem: from Continuous to Discrete." In *Time Warps, String Edits, and Macromolecules – The Theory and Practice of Sequence Comparison,* chap. 4.

Levé, Florence, Richard Groult, Guillaume Arnaud, Cyril Séguin, Rémi Gaymay, and Mathieu Giraud. 2011. "Rhythm extraction from polyphonic symbolic music." In *International Society for Music Information Retrieval Conference (ISMIR 2011),* 375–380.

Mongeau, Marcel, and David Sankoff. 1990. "Comparison of Musical Sequences." *Computers and the Humanities* 24 (3): 161–175.

Needleman, Saul B., and Christian D. Wunsch. 1970. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of Molecular Biology* 48 (3): 443–453.

Pachet, François, Alexandre Papadopoulos, and Pierre Roy. 2017. "Sampling variations of sequences for structured music generation." In *International Society for Music Information Retrieval Conference (ISMIR 2017),* 167–173.

Smith, T. F., and M. S. Waterman. 1981. "Identification of common molecular subsequences." *Jouranl of Molecular Biolology* 147: 195–197.

Tremblay, Guy, and France Champagne. 2002. "Automatic Marking of Musical Dictations By Applying the Edit Distance Algorithm On a Symbolic Music Representation." In *First Int. Conference Musical Application Using XML (MAX 2002),* .

Ukkonen, Esko. 1985. "Algorithms for Approximate String Matching." *Information and Control* 64: 100–118.

Ukkonen, Esko, Kjell Lemström, and Veli Mäkinen. 2003. "Geometric algorithms for transposition invariant content based music retrieval." In *International Conference on Music Information Retrieval (ISMIR 2003),* 193–199.

Velarde, Gissel, Tillman Weyde, and David Meredith. 2013. "An approach to melodic segmentation and classification based on filtering with the Haar-wavelet." *Journal of New Music Research* 42 (4): 325–345.