Comparison of Musical Sequences

Marcel Mongeau

Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

and

David Sankoff

Centre de recherches mathématiques, Université de Montréal, C.P. 6128, Succursale A, Montréal, Québec H3C 3J7, Canada

Abstract: Concepts from the theory of sequence comparison are adapted to measure the overall similarity or dissimilarity between two musical scores. A key element is the notion of consolidation and fragmentation, different both from the deletions and insertions familiar in sequence comparison, and from the compressions and expansions of time warping in automatic speech recognition. The measure of comparison is defined so as to detect similarities in melodic line despite gross differences in key, mode or tempo. A dynamic programming algorithm is presented for calculating the measure, and is programmed and applied to a set of variations on a theme by Mozart. Cluster analysis and spatial representation of the results confirm subjective impressions of the patterns of similarities among the variations. A generalization of the algorithm is presented for detecting locally similar portions in two scores, and is then applied.

Key Words: sequence comparison, dynamic programming, musical pattern recognition, melodic line

Marcel Mongeau obtained his B.Sc. and M.Sc. degrees at the Université de Montréal and is currently completing his doctorate at the University of Waterloo.

David Sankoff (Ph.D., McGill) is a Professor in the Département de mathématiques et statistique and is also attached to the Centre de recherches mathématiques at the Université de Montréal. His research intersts include sociolinguistics — specifically the quantitative approach inherent in linguistic variation theory — statistical classification theory, biomathematics and computational biology — particularly algorithms for macromolecular sequence analysis and the reconstruction of phylogenetic trees.

1. Introduction

Much work in the automated analysis of melodic line is concerned with the comparison of variants of a tune or the detection of patterns in a tune where corresponding melodic fragments may be transformed or distorted in various ways. For example, Stech (1981) searches for versions of a given pattern allowing 26 predefined types of melodic modification. These consist of different combinations of rhythmic retrograde, tonal retrograde and inversion, and permit arbitrary transposition and location within the measure. Dillon and Hunter (1982) provide for matching patterns on the basis of stressed pitches only, disregarding differences in the rest of the measure.

In this paper, rather than establishing which aspects of a melodic pattern must be fixed or setting up a class of permitted transformations of fixed patterns, we define a distance between any two melodies which depends on the tonal contour and the rhythmic structure. This approach avoids the arbitrariness of predefining the class of transformations or distortions which are to be permitted, and the subjectivity of deciding on "the

Computers and the Humanities **24**: 161–175, 1990.

boundary between a tune and its true variants . . . and tunes which are similar in one or another respect but are not variants" (Dillon and Hunter, 1982).

This more general type of distance measure, which is appropriate to variable-length linear structures such as tunes or shorter musical patterns, may require costly and complex calculations, so we will concentrate here on the elaboration of relatively efficient algorithms to carry out this computation.

One of the advantages of a distance-based approach is that, given a set of several melodies, they may be classified in a hierarchical manner (cf. Logrippo and Stepien, 1986) or as a configuration of points in a metric space. Hierarchical clustering leads to a classification which is more than just a set of disjoint types; it permits major classes subdivided into subclasses, which can be further subdivided, and so on. A spatial configuration, on the other hand, can often be projected into a two- or three-dimensional subspace which conserves much of the pattern of proximity and distance in the original data, allowing for easy visualization and identification of the major dimensions of variability among the melodies.

2. Sequence Comparison

Traditional ways of comparing sequences work only if the sequences have the same length n; each term from one sequence is compared to the term in the same position in the other sequence and the similarities or dissimilarities are summed over all of these pairs of corresponding terms. In the modern theory of finite sequence comparison (cf. Sankoff and Kruskal, 1983), the general approach is the search for an *optimal* correspondence between the elements of the two sequences (which have not necessarily the same length) among every possible correspondence satisfying some conditions such as the preservation of the order of the elements.

A natural way to quantify the difference between two sequences $A = a_1, a_2, \ldots, a_m$ and $B = b_1, b_2, \ldots b_n$ is to count the minimal number of transformations (chosen among a predetermined set of allowed transformations) which must be applied to the first sequence in order to obtain the second one. This number is called the *dissimilarity.*

Typical transformations include *deletion* of a term from a sequence, *insertion* of a term between two preexisting terms in a sequence and *replacement* of one term by another. To illustrate, the minimal number of transformations necessary to change the sequence A = a, b, c, d, e, f, g to B = a, c, d, e, f, h, i is three; for example, deletion of 'b,' insertion of 'i' and replacement of 'g' by 'h,' the other terms remaining the same.

The *trace* of a series of transformations from A to B, as illustrated in Figure 1, consists of the *source sequence* A, the *target sequence* B and lines linking replaced elements and unchanged elements. When transformations must be deletions, insertions or replacements only, a sequence element can touch no more than one line and the lines do not cross each other.

These concepts can be generalized by associat-

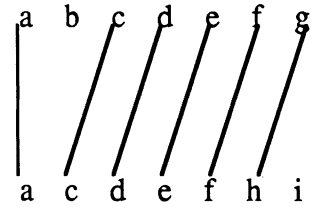


Figure 1. Trace linking two sequences.

ing a weight to each type of transformation. Thus, we do not necessarily count one for each transformation but a predefined value related to the kind of transformation (deletion, insertion or replacement), and to the elements involved in the transformation. The dissimilarity becomes the minimum sum of weights among all possible series of transformations converting sequence A to B. It may be computed rapidly (in time proportional to mn) by recurrence. Let d_{ij} be the dissimilarity between a_1, a_2, . . . , a_i and b_1, b_2, . . . , b_j. The recurrence equation for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$ is

$$d_{ij} = \min \begin{cases} d_{i-1,j} + w(a_i, \varnothing) \\ d_{i-1,j-1} + w(a_i, b_j) \\ d_{i,j-1} + w(\varnothing, b_j) \end{cases}$$

where $w(a_i, \varnothing)$ is the weight associated with the deletion of a_i, $w(\varnothing, b_j)$ with the insertion of b_j and $w(a_i, b_j)$ with the replacement of a_i by b_j. We assume $w(a_i, b_j) = 0$ if $a_i = b_j$. The initial conditions are

$$d_{i0} = d_{i-1,0} + w(a_i, \varnothing), i \geqslant 1;$$

$$d_{0j} = d_{0,j-1} + w(\varnothing, b_j), j \geqslant 1$$

and

$$d_{00} = 0,$$

since if one of the sequences has no element, the only possible transformations are deletions or insertions respectively.

This recurrence may be applied to all pairs (i, j) in any order, as long as it has already been applied to $(i-1, j)$, $(i, j-1)$ and $(i-1, j-1)$ before being applied to (i, j). In order to obtain the optimal trace linking two sequences we construct, during the calculation of each d_{ij}, pointers indicating which of the right hand side terms achieves the minimum. The best alignment and trace are obtained by following pointers backwards from the final dissimilarity d_{mn}. It is not difficult to prove that this calculation yields the minimum sum of weights among all possible series of transformations converting sequence A to B, together with an optimal trace linking the two sequences.

Note that if $w(x, y) = w(y, x)$ for all terms x and y (including ø), the calculation of d_{mn} between A and B gives the same result as the calculation of d_{nm} between B and A, with deletion in the first case becoming insertion in the second, and vice versa.

These ideas are fundamental to the mathematical theory of sequence comparison, which has found many applications in science and the humanities.

3. Musical Score as a Sequence

Any monophonic score may be regarded as a series of ordered pairs with the pitch of the note as the first component and its length as the second. Rests are indicated with a dummy symbol in the first component. In order that our analysis remain invariant under transformation from one key to another, and from one tempo to another, we code the pitch of each note according to how far it is, e.g. in semitones, from the tonic. The length of the notes are coded in sixteenth note values.

4. Transformations and Weights

The assignment of weights to transformations must be sensitive to the kind of musical differences we wish to measure. In this study we wish to transcend gross differences in overall key signature, tempo or mode between two scores, since these are obvious without any quantitative assessment. Rather, we are interested in local differences in rhythm and melody. In the case of a replacement the weight is the sum of two quantities: $w(a_i, b_j) = w_{interval}(a_i, b_j) + k_1 w_{length}(a_i, b_j)$, where k_1 represents the relative contribution of length difference versus that of pitch difference. $w_{interval}(a_i, b_j)$ is a predefined value determined by the relative position of the notes a_i and b_j (recalling that our notation effectively transposes each musical sequence to a common scale), multiplied by the length of the shorter of a_i and b_j, so that $w_{interval}$ is reduced if a short note such as grace-note is involved. We define $w_{length}(a_i, b_j)$ simply as the difference of the lengths of the two notes. For tonal music, interval weight is linked to the consonance of the interval. For example, a small weight is attributed to the replacement of a note by one an octave or a fifth above, since these are consonant intervals (in each case the two notes have a simple frequency ratio: 2/1 and 3/2 for the octave and the fifth respectively) whereas $w_{interval}$ will be bigger for dissonant intervals, such as a

second or a seventh (ratios of 9/8 and 15/8 with the tonic).

To assure that interval weight is independent of both key and mode, it is helpful to convert the pitch of a note to degrees of the scale from the tonic. For example, if the source sequence is in G major and the target one in G minor we assign weight zero for the replacement of a B (fifth semitone from the tonic) by a B flat (fourth semitone from the tonic) since both notes are the third scale-degree of their respective scale (cf. Figure 2). Otherwise the weight for this interval would be inappropriately large, a semitone being a dissonant interval.

Denote deg(n) the weight associated to pairs of notes which differ by n degrees of scale. A special calculation is needed if one of the notes being compared is not in its scale (i.e. it is not a degree of the scale of the score it is in). In this case, the interval weight is denoted by ton(m), is computed from the difference m in semitones and is higher than for neighbouring intervals computed from degrees (since it is less consonant): we will assume that the relationship is approximately of form $\text{ton}(m) = \mu\,\text{deg}(n(m)) + \partial$, where n(m) is the neighbouring interval computed by degrees and μ and ∂ are parameters to be determined. For example, n(7) = 4 since the fifth (interval of 4 degrees) consists of 7 semitones; n(4) = 2 for a major scale and n(3) = 2 for a minor scale, since the third (interval of 2 degrees) consists of 4 or 3 semitones depending on if the third is major or minor, and so on. If both notes are not in their respective scale, the interval weight is computed from the difference of degrees after subtracting (or adding) one semitone to both notes, except when that is not possible. (If the two notes are the tenth and eleventh semitones of minor scales cf. Figure 2; in this case we use ton(m) and the μ and ∂ parameters.) These considerations of degrees and scales are not pertinent in an atonal music context where w might be, for example, constant over all intervals.

We consider $w(a_i, \varnothing)$, the weight associated with a deletion, to be a special case of $w(a_i, b_j)$, since a deletion is, in effect, as the replacement of a_i by a note of length zero. Appropriately, the term $w_{interval}$ is reduced to zero since the smaller of the length of a_i and zero is zero, so that $w(a_i, \varnothing)$ is simply the length of the deleted note a_i times k_1. In the same way, $w(\varnothing, b_j)$ is the length of the inserted note b_j times k_1.

5. Analogies with Time-Warping

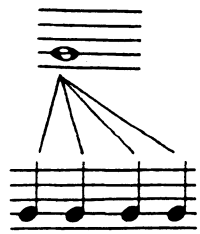
It is necessary in the comparison of musical sequences to allow two more kinds of transformation, similar to the compressions and expansions used in time-warping for automated speech re-

Major scale degrees	1		2		3	4		5		6		7
Minor scale degrees	1		2	3		4		5	6			7
Semitones	1	2	3	4	5	6	7	8	9	10	11	12
Difference of degrees with the tonic	0		1	2	2	3		4	5	5		6
Difference of semitones with the tonic	0	1	2	3	4	5	6	7	8	9	10	11

Figure 2. Degrees of major and minor scales in the chromatic scale.

cognition (cf. Kruskal and Liberman, 1983). These new transformations which we call *consolidations* and *fragmentations*, involve the replacement of several elements by a single one and the replacement of one element by several, respectively. To motivate this, we note that, for example, a whole note replaced by four quarter notes of the same pitch should not be very costly. Were we constrainted to insertions, deletions and replacements, however, it would cost at least one replacement and three insertions. It seems advantageous to be able to count it as a single transformation: a fragmentation.

In the same way as for a replacement, the weight associated with a fragmentation or a consolidation is a linear combination of w_{length} and $w_{interval}$. For a fragmentation, $w_{interval}$ is the sum of the $w_{interval}$ s between each replacing note and the replaced one while w_{length} is the difference between the total length of the replacing notes and the length of the replaced one. For example in the following fragmentation, since the length of the whole note is equal to the total length of the four replacing notes, the weight associated to the total difference of length is zero. $w_{interval}$ will be the sum of the $w_{interval}$ s corresponding to each of the trace lines. Interval weight and total length difference weight are defined in a similar way for a consolidation.



The consolidation/fragmentation concept resembles that of compression/expansion, with the essential difference that the latter, mainly used in the processing of human speech, is applicable to sequences resulting from sampling at regular intervals of a continuous function, usually of time. Compressions and expansions are local distortions or "warpings" of the time axis, the sequences they link are discretizations of two continuous functions whose trajectories traverse (approximately) the same curve in feature space in the same direction though at possibly very different rates. In the context of musical sequences, however, the melody function is essentially discrete, the time interval between two elements is not regular and in consolidation and fragmentation a note is most often compared to a series of notes with the *same* total time value (through the effect of w_{length}) as opposed to compression and expansion.

6. The Algorithm

By integrating consolidation and fragmentation into the set of transformations and letting the weight associated with a replacement be a linear combination of $w_{interval}$ and w_{length}, the dissimilarity recurrence equation becomes

$$d_{ij} = \min \begin{cases} d_{i-1,j} + w(a_i, \varnothing) & \text{(deletion)} \\ d_{i,j-1} + w(\varnothing, b_j) & \text{(insertion)} \\ \{d_{i-1,j-k} + w(a_i, b_{j-k+1}, \ldots, b_j), \\ 2 \leqslant k \leqslant j\} & \text{(fragmentation)} \\ \{d_{i-k,j-1} + w(a_{i-k+1}, \ldots, a_i, b_j), \\ 2 \leqslant k \leqslant i\} & \text{(consolidation)} \\ d_{i-1,j-1} + w(a_i, b_j) & \text{(replacement)} \end{cases}$$

for $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$, where $w(a_i, b_{j-k+1}, \ldots, b_j)$ and $w(a_{i-k+1}, \ldots, a_i, b_j)$ are the predefined weights associated to a fragmentation and a consolidation respectively. Initial conditions are

$$d_{i0} = d_{i-1,0} + w(a_i, \varnothing), i \geqslant 1 \quad \text{(deletion);}$$

$$d_{0j} = d_{0,j-1} + w(\varnothing, b_j), j \geqslant 1 \quad \text{(insertion)}$$

and

$$d_{00} = 0,$$

since at this stage of the recurrence, deletions and insertions are respectively the only possible transformations, one of the sequences having no terms. Under this new definition of dissimilarity, a typical trace diagram might be as in Figure 3.

Whereas the algorithm presented in section 2 required computer time proportional to n^2 (in the case $m = n$), this new algorithm might seem to necessitate a time of order n^3 since for each of the n^2 values of d_{ij} to be computed ($1 \leqslant i, j \leqslant n$) we have to find minimum not among three possibilities as before, but among $i + j + 1$, which is bounded only by $2n + 1$. However, the algorithm may be adapted to require only order n^2 computer

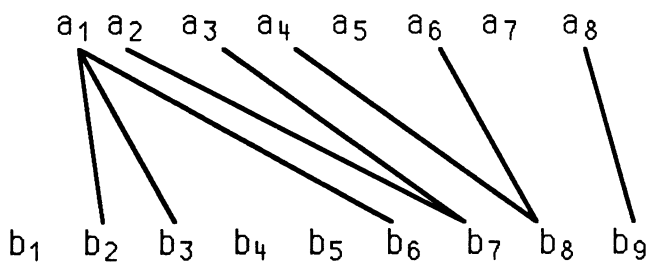


Figure 3. Trace involving consolidations and fragmentations.

time. This involves the use of two constants F and C such that the recurrence equation

$$d_{ij} = \min \begin{cases} d_{i-1,j} + w(a_i, \varnothing) & \text{(deletion)} \\ d_{i,j-1} + w(\varnothing, b_j) & \text{(insertion)} \\ \{d_{i-1,j-k} + w(a_i, b_{j-k+1}, \ldots, b_j), & \\ 2 \leqslant k \leqslant \min(j, F)\} & \text{(fragmentation)} \\ \{d_{i-k,j-1} + w(a_{i-k+1}, \ldots, a_i, b_j), & \\ 2 \leqslant k \leqslant \min(i, C)\} & \text{(consolidation)} \\ d_{i-1,j-1} + w(a_i, b_j) & \text{(replacement)} \end{cases}$$

is equivalent to the previous one. This is the case if it is never necessary to consider fragmentations of a_i into more than F elements or to consider the consolidations of more than C elements into b_j. For example we can always find an F such that it is more worthwhile to insert a number of terms than to extend a fragmentation to more than F elements. We simply choose F to be the smallest integer larger than or equal to the ratio $\max\{L(a_p), 1 \leqslant p \leqslant m\}/\min\{L(b_q), 1 \leqslant q \leqslant n\}$, where $L(a_p)$ and $L(b_q)$ are the lengths of the notes a_p and b_q since the fragmented note will clearly be shorter than the total length of the F fragments and hence, extending the fragmentation to more than F elements would increase the w_{length} component of the weight. In practice, when computing a given d_{ij} and considering successively fragmentations into 2, 3, 4, ... elements, it is not even necessary to continue until the fragmentation of a_i into F elements; we can stop when the total length of the fragment notes becomes equal to or larger than that of a_i, the fragmented note. Similar remarks apply to C.

Again, it can be shown that the optimal trace linking two musical sequences is obtained through the use of pointers for each d_{ij} of the previous recurrence equation and the best alignment is found by following the pointers starting at d_{mn}.

Of course, since F and C depend on the ratio of note lengths in the two scores, the algorithm can only be considered strictly of order n^2 over some set of scores (or possible scores) in which these quantities are bounded. Practically speaking, this is always the case.

It should be noted that Logrippo and Stepien (1986) also treated scores as abstract sequences, but confined themselves to pitch considerations in making alignments, and without allowing for fragmentation, consolidation or other time-warp analogies. They did not envisage recourse to sequence alignment algorithms for optimizing alignments, though they did suggest weighting matches according to rhythmic considerations.

7. Parameter Determination

In order to apply the algorithm, the values of the parameters involved in the dissimilarity recurrence equation have to be given. These parameters are k_1, the relative contribution of w_{length} versus

$w_{interval}$; deg(n), n = 0, 1, . . . , 6, the weight associated with an interval of n degrees; μ and ∂, the parameters involved in the equation used to compute ton(m) from the weights associated with intervals computed by degrees, and *rest*, which is the value given to $w_{interval}$ when one of the notes involved in the transformation (replacement, fragmentation or consolidation) is a rest (when two rests are matched, $w_{interval}$ takes the same value as in the case of two notes of same pitch). Interval weights are determined modulo an octave. For example, the weight for an octave-and-a-third is the same as for a third. Interval parameters are determined according to consonance. We will assign increasing weights to intervals in order of increasing dissonance: unison, fifth, third, sixth, fourth, seventh and second. Despite the fact that the fourth has a simpler frequency ratio than do the third and the sixth, 4/3 compared to 5/4 and 5/3 respectively, the third and the sixth are traditionally considered more consonant than the fourth in occidental music. For the applications in this paper, we chose the values:

deg(0) = 0 (unison (identity replacement), octave, two octaves, . . .);
deg(4) = 0.1 (fifth, octave and a fifth, . . .);
deg(2) = 0.2 (third, octave and a third, . . .);
deg(5) = 0.35 (sixth, . . .);
deg(3) = 0.5 (fourth, . . .);
deg(6) = 0.8 (seventh, . . .);
deg(1) = 0.9 (second, . . .);
rest = 0.1;
μ = 2;
∂ = 0.6;

and hence we get for the other weights, using the formula ton(m) = μdeg(n(m)) + ∂ (raising the weights computed by degrees):

ton(0) = 0.6 (unison, octave, . . .);
ton(7) = 0.8 (perfect fifth, octave and a perfect fifth, . . .);
ton(4) = 1 (major third, . . .);
ton(3) = 1 (minor third, . . .);
ton(9) = 1.3 (major sixth, . . .);
ton(8) = 1.3 (minor sixth, . . .);
ton(5) = 1.6 (perfect fourth, . . .);
ton(6) = 1.8 (augmented fourth, . . .);
ton(10) = 2.2 (minor seventh, . . .);
ton(2) = 2.3 (major second, . . .);
ton(11) = 2.5 (major seventh, . . .);

and

ton(1) = 2.6 (minor second, . . .).

We note in the values listed above that ton(6), ton(11) and ton(1) are slightly larger than stipulated by the equation ton(m) = μdeg(n(m)) + ∂, and ton(2) is slightly less, the augmented fourth being less consonant than the perfect fourth, the major seventh less than the minor and the minor second much less than the major second. Finally we assigned a weight to the rest of exactly the same value as that of the most consonant interval (other than the identity replacement). We can be really confident only about the order of weights in terms of decreasing consonance. The precise values we have calculated are debatable, or could be optimized with respect to some data set. In any case, some assignment of parameter values is necessary to evaluate the performance of the algorithm.

Once the weights associated to the intervals are fixed, it is relatively easy to determine the only remaining parameter, k_1, which represents the relative contribution of w_{length} and $w_{interval}$. We proceed heuristically on pairs of musical sequences which are known to be quite similar since in this case we know roughly what to expect for the form of the trace diagram.

Mozart's nine variations on the theme: Ah! vous dirai-je, maman (K. 300, "Twinkle, Twinkle Little Star," arrangement in Duschesnes [1962]) constitute an ideal collection of such musical sequences to start with, they are of comparable lengths: the theme and most of the variations have twelve 4/4 bars. Variation 6 has twenty-four 3/4 bars and Variation 4 twelve 6/8 bars, so that by multiplying the length of each note of the former by 2/3 and each note of the latter by 4/3, all the sequences are endowed with the same total time.

To determine k_1 we will study its effect on the traces linking three pairs of sequences. The first pair involves two very similar sequences: the theme and Variation 5, the former in a major key, the latter in a minor one. The second pair consists of two quite similar sequences, Variation 2 and 3, and the third, two relatively dissimilar musical sequences: Variations 3 and 7. If for a value of k_1

the traces align too often two highly dissonant notes of same length, we may suspect that the differences of length are too strongly penalized compared to differences of pitch, i.e. k_1 is too high. Inversely, when we notice that consolidations and fragmentations tend to link a single note of a certain length with a group of notes whose total length is very different simply because all the notes of the group are consonant with the single note, k_1 must be too low. As another way of assessing a value of k_1, we may observe if the bar lines divide trace diagrams into twelve bar lines (the bar line information is not given in the input to the algorithm). If few trace lines cross bar divisions we consider the trace favorably.

Based on these criteria, we obtain the traces in Figures 4, 5 and 6 with the final value: $k_1 = 0.348$. We notice that the bar alignment criterion is only partially met between Variations 3 and 7, is largely met by Variations 2 and 3, and fully met between the theme and Variation 5. (If we connect corresponding bar lines we see that no trace lines cross them.) This last trace confirms the mode independence of the algorithm since the theme is in major key while Variation 5 is in a minor one.

That the set of parameters responsible for the traces is reasonable, is confirmed by examining Figures 4 and 5, where notes having obviously analogous roles in the two melodies are linked in the trace. Inappropriate parameter values inevitably miss some of these correspondences.

8. Patterns of Similarity among Variations on a Theme

We apply our musical comparison algorithm, with parameters as determined in the preceding section, to the set of all possible pairs of these variations. Prior to further quantitative analysis, we sketch a "subjective" assessment of the different variations. Variation 1 is almost identical to the theme, with only a few fragmentations of the original quarter notes into eighth notes. Variation 2 contains many more such fragmentations. Variation 3 carries this rhythmic complication much further, introducing sixteenth notes but still following the melodic structure of the theme rather

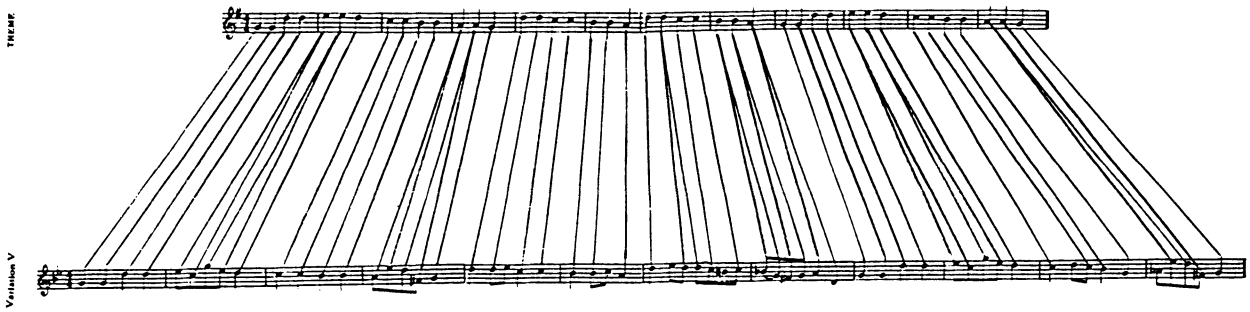


Figure 4. Trace linking Variation 5 to Theme.

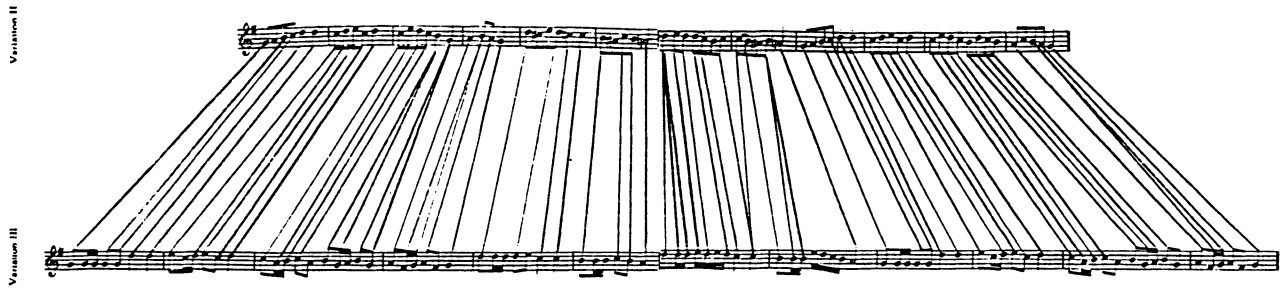


Figure 5. Trace linking Variation 2 to Variation 3.

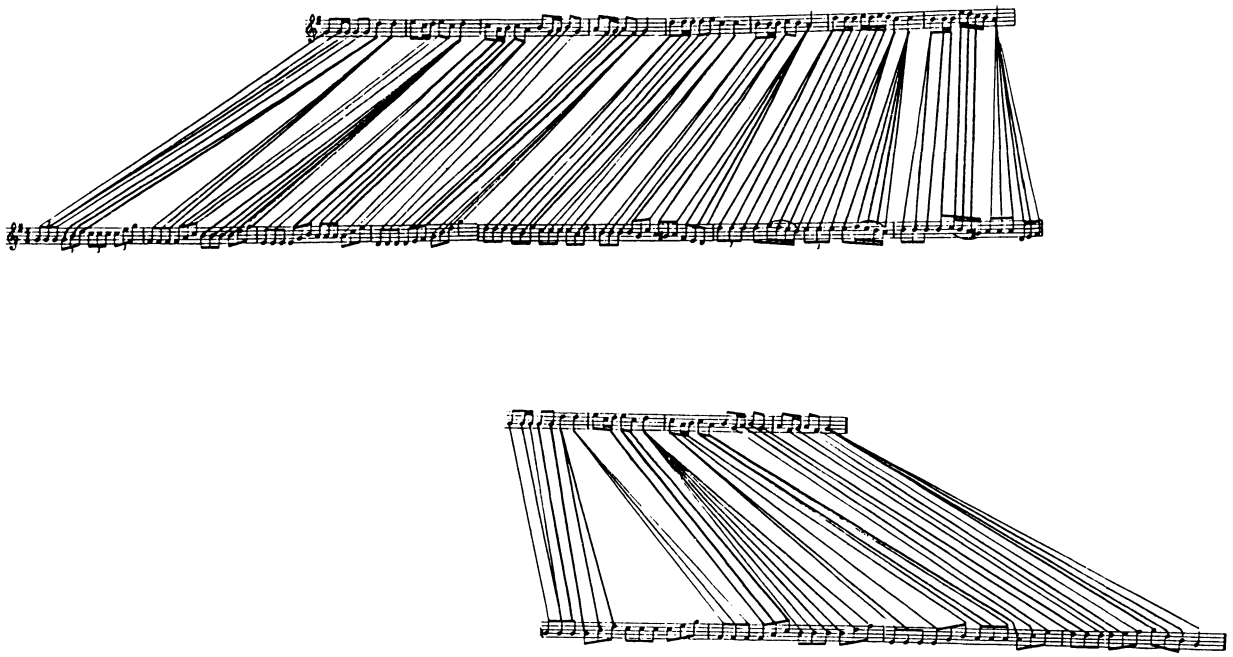


Figure 6. Trace linking Variation 3 to Variation 7.

strictly. Likewise for Variation 4 which contains many grace notes and smoothly ascending and descending diatonic and arpeggio passages. Variation 5 returns closer to the basic melody, but is in a minor key, as is Variation 6. The rhythmic complexity of Variation 6 disguises the theme, but it is still discernible, as opposed to the remaining three variations in which there is very little identifiable of the original melody. Variation 7 consists mostly of arpeggio triplets while Variation 8 and 9 are made up of long passages of sixteenth notes. In Variation 8, these passages contain rather large intervals and few smoothly ascending or descending sequences, while the opposite holds for Variation 9.

If we apply our algorithm to every pair of variations, we obtain the dissimilarities shown in Table 1. We note first the central role of the theme: with some exceptions it is generally more similar to each variation than the latter is to any other variation. In order to visualize the overall patterns contained in the array, a rough classifica-

Table 1. Dissimilarities between each pair of variations.

	Theme	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9
Theme	0									
Var1	19.1	0								
Var2	29.1	29.1	0							
Var3	17.2	22.7	26.6	0						
Var4	33.1	33.3	33.2	33.5	0					
Var5	17.6	20.0	30.3	21.0	34.0	0				
Var6	33.1	41.8	44.1	39.1	46.7	34.8	0			
Var7	40.5	41.7	49.5	45.1	44.5	45.0	40.3	0		
Var8	41.5	43.9	44.4	39.7	42.1	40.1	47.4	54.4	0	
Var9	45.1	48.3	49.2	38.1	45.9	46.8	44.7	49.3	38.0	0

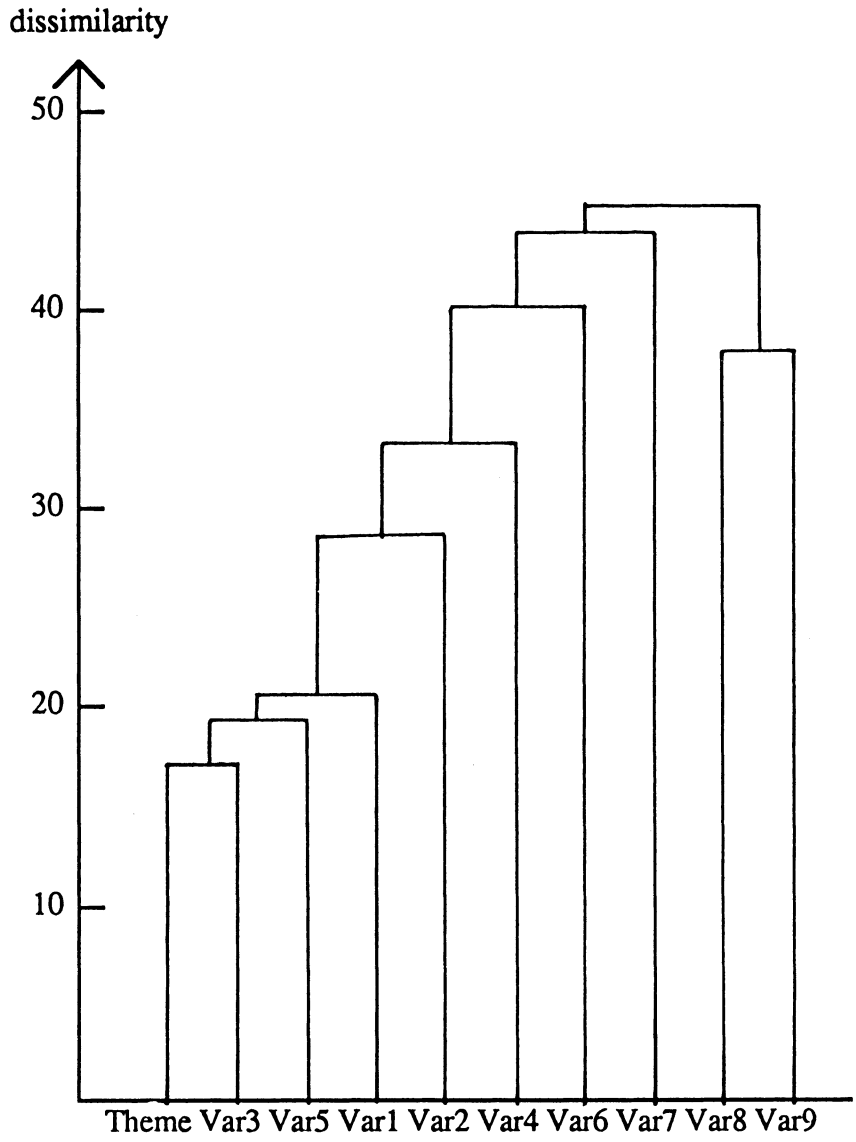


Figure 7. Cluster analysis. Average link method.

tion of the nine variations and the theme can be made through cluster analysis. The tree in Figure 7 is derived using the average-link method (Williams and Lance, 1977). It appears from this tree that variations in which the theme can easily be recognized (Variations 1, 3 and 5) group together (with the theme) rather closely. Likewise Variations 8 and 9, which have almost identical rhythmic structures, group together. Finally, we should note the proximity of variations in a major key with Variation 5 despite the fact that the latter is in a minor one.

A different way of visualizing the data is through a spatial representation as given by principal components analysis (cf. Rao, 1965). Our dissimilarity is not however, a metric. For example if one takes a and b a fifth apart, and c a fifth apart from b (highly consonant intervals), then the interval between a and c is an octave and a second (dissonant interval). In other words, the

interval weights do not satisfy the triangular inequality, since for this example $w_{interval}(a, b) + w_{interval}(b, c) < w_{interval}(a, c)$. Hence, neither does the dissimilarity measure. Moreover we do not have $w_{interval}(x, y) = 0$ iff $x = y$ when for example x and y are two notes one octave away from each other (the weight associated with this interval having been fixed to zero). Nevertheless it does not deviate from data-analytic practice to assume that for long enough melodies d approximates a metric. This will enable us through the use of principal components analysis to represent each variation as a point on the plane.

The best representation (preserving a maximum of the variability in the data) of ten points (the nine variations and the theme) in the plane, that is, using the first two principal components as axes, which together represent more than 54% of the total variation in the data, yields the configuration in Figure 8. The same remarks made about the results obtained with cluster analysis apply here. We note, however, that the principal components analysis makes a clearer separation between dissimilar variations than does the cluster analysis, and that the internal structure of the more theme-like group is less well resolved. If we consider the first three principal axes, which represent nearly 70% of the total variation, we obtain some refinement in the main group. The central role of the theme is confirmed and the distance separating Variations 6 and 7 is clearer.

9. General Case: Seeking for Local Similarities

The algorithm studied in the previous sections is meaningful for comparing two scores in their entirety. Often, however, especially when two

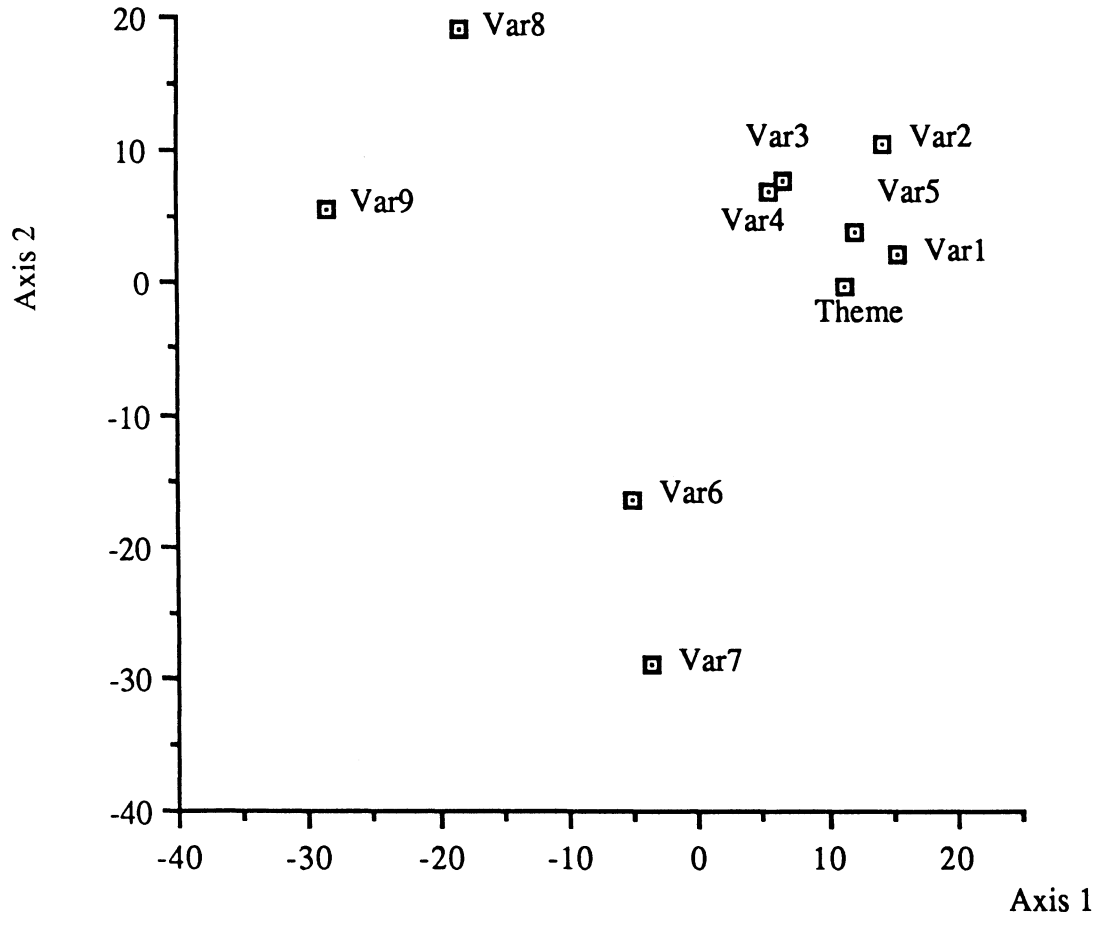


Figure 8. Principal components analysis.

scores are of very different lengths, there is no motivation for such a global comparison. Instead, we may wish to detect and measure similar portions of two scores. We now introduce an algorithm for doing this which is almost identical to that presented in the previous sections. The addition of a single parameter controlling the length of the similar portions to be found is enough to generalize the previous method.

Kruskal and Sankoff (1983), in order to find local alignments (traces between portions of two sequences) in molecular biology, propose a modification of an algorithm first suggested by Smith and Waterman (1981). If we were to search for portions of two sequences with minimum distance as defined in the basic sequence comparison algorithm, this minimum would always be achievable with a meaningless alignment of very short portions (such as alignment of two single elements). Instead, we replace the dissimilarity used in the algorithm of Section 2 by an alignment "quality" function to be maximized, involving both positive and negative values, depending on the type of transformation. Let q_{ij} be the maximum quality of any local alignment between portions ending at a_i and b_j. If all such alignments have negative quality then the maximum is the null alignment and $q_{ij} = 0$. Since every local alignment between $a_1, a_2, \ldots, a_m$ and $b_1, b_2, \ldots, b_n$ contains some final a_i and final b_j, the maximum quality possible is $\max\{q_{ij}: 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$ where q_{ij} is computed with the following recurrence:

$$q_{ij} = \max \begin{cases} q_{i-1,j} + s(a_i, \varnothing) \\ q_{i-1,j-1} + s(a_i, b_j) \\ q_{i,j-1} + s(\varnothing, b_j) \\ 0 \end{cases}$$

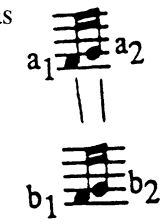
where $q_{i0} = 0$ for all i, $q_{0j} = 0$ for all j and $s(a_i, \varnothing) < 0$, $s(\varnothing, b_j) < 0$ and $s(a_i, b_j)$ are the values corresponding to deletions, insertions and replacements, respectively. Note that $s(a_i, b_j) > 0$ if $a_i = b_j$.

Once the q_{ij} are calculated, the best local alignment is obtained by making use of pointers calculated at the same time as the q_{ij} indicating, for each q_{ij}, which of the right hand side terms was responsible for the maximization. The traceback path starts with the maximizing q_{ij} and ends when a nil q_{ij} value is met.

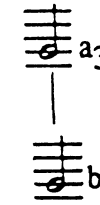
In some applications we look not only for the best local alignment but also for the second "most similar" pair and so on, and generally we do not want these to be merely subsets of portions already found. Smith and Waterman suggest eliminating as possible endpoints for an alignment all q_{ij} encountered in the traceback from the optimal q_{ij} down to the beginning of this optimal alignment and then looking for the second optimal quality amongst remaining q_{ij}. But proceeding this way will still generally lead to variants of the first alignment found. The modification suggested by Kruskal and Sankoff is to label as 'used' each (i, j) encountered in the trace of the optimal alignment found and to reapply the algorithm with the difference that we let $q_{ij} = 0$ for all q_{ij} labeled 'used'. Once all the q_{ij} are re-computed, we take as second best alignment the maximal q_{ij} value. We repeat this procedure in order to obtain the other best alignments.

In the same way as we generalized the full-sequence comparison algorithm to the comparison of musical scores, we adapt the algorithm for finding similar portions of two sequences described here to finding similar portions of two scores. Considering a score as a series of ordered pairs (pitch and length) and allowing consolidations and fragmentations, we first construct the quality function q_{ij} exactly in the same way as d_{ij} (dissimilarity), except that each weight is replaced by (k_2 − weight), where k_2 is a parameter defined so that (k_2 − weight) is negative for insertion and deletion and positive for at least identity replacements.

This approach is not completely satisfying since it awards a high score to two replacements involving short notes such as



and a low weight to a replacement involving two long notes such as



(the first weight is $2k_2$ whereas the second is k_2). In order to avoid this problem we multiply k_2 by the total length of the notes involved in the transformation considered. Thus, the scores associated with deletion, replacement and fragmentation will be respectively:

$$s(a_i, \varnothing) = k_2 L(a_i) - w(a_i, \varnothing),$$

$$s(a_i, b_j) = k_2(L(a_i) + L(b_j)) - w(a_i, b_j)$$

and

$$s(a_i, b_{j-k+1}, \ldots, b_j) = k_2\Big(L(a_i) + \sum_{s=j-k+1}^{j} L(b_s)\Big) - - w(a_i, b_{j-k+1}, \ldots, b_j)$$

where $L(a_i)$ and $L(b_j)$ are the lengths of the notes a_i and b_j. Thus the quality recurrence equation is

$$q_{ij} = \max \begin{cases} q_{i-1,j} + k_2 L(a_i) - w(a_i, \varnothing) \\ q_{i,j-1} + k_2 L(b_j) - w(\varnothing, b_j) \\ \Big\{q_{i-1,j-k} + k_2\Big(L(a_i) + \sum_{s=j-k+1}^{j} L(b_s)\Big) - \\ \quad - w(a_i, b_{j-k+1}, \ldots, b_j), \\ \quad 2 \leqslant k \leqslant \min(j, F)\Big\} \\ \Big\{q_{i-k,j-1} + k_2\Big(\sum_{s=i-k+1}^{i} L(a_s) + \\ \quad + L(b_j)\Big) - w(a_{i-k+1}, \ldots, a_i, b_j), \\ \quad 2 \leqslant k \leqslant \min(i, C)\Big\} \\ q_{i-1,j-1} + k_2(L(a_i) + L(b_j)) - w(a_i, b_j) \\ 0 \end{cases}$$

for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$, with initial conditions:

$$q_{i0} = \max \begin{cases} q_{i-1,0} + k_2 L(a_i) - w(a_i, \varnothing) \\ 0 \end{cases}, \quad i \geqslant 1;$$

$$q_{0j} = \max \begin{cases} q_{0,j-1} + k_2 L(b_j) - w(\varnothing, b_j) \\ 0 \end{cases}, \quad j \geqslant 1$$

and

$$q_{00} = 0$$

Again, to obtain all the good local alignments, we trace back from all the highest values of q_{ij} and stop whenever a nil q_{ij} value is met.

The parameter k_2 can be interpreted in terms of length of the local alignments, this length being proportional to k_2. Moreover, we can prove that the algorithm which seeks for local similarities is a generalization of the algorithm for entire sequences discussed in section 6. Indeed, if k_2 is chosen large enough, the optimal local alignment is the alignment obtained with the algorithm of section 6 (and the portions found are thus the entire source and target sequences).

10. Seeking for Local Similarities in Mozart's Alleluja

In applying the algorithm of section 9 to the piece Alleluja (K165, 1773), the score (Anonymous 1932) was used both as source and target sequence. First, we labeled as 'used' all (i, i), $1 \leqslant i \leqslant m$ in order to avoid finding, as first optimal local alignment, the trivial identity alignment (between the whole piece and itself).

The twelve best local alignments (with $k_2 = 0.05$) consist almost entirely of identity replacements: eight are properly identity alignments such as the eighth best local alignment shown in Figure 9a, which does, however, reveal an interesting feature: the portion reoccurs twice, as in a canon, with a two bar delay. The four other local alignments each contains one or two notes which differ only in their length or their pitch or one consolidation of two notes, as in the third best pair of similar portions, shown in Figure 9b.

11. Discussion

The automated analysis has served to confirm and quantify our subjective impressions of the dimensions of variability in the variations. At the same time this exercise is a validation of the concepts and the actual parameter values used in the comparison algorithm.

The algorithm we have elaborated is valid in the context of any monophonic tonal music, although the parameter values were determined using a restricted set of variations on a theme. These variations are diverse enough to require trace diagrams exemplifying a wide range of transformations between scores. The addition of a single parameter which has an influence on the length of the portions found was sufficient to generalize the algorithm into one designed for detecting similar portions in two scores.

The algorithm for comparing entire sequences

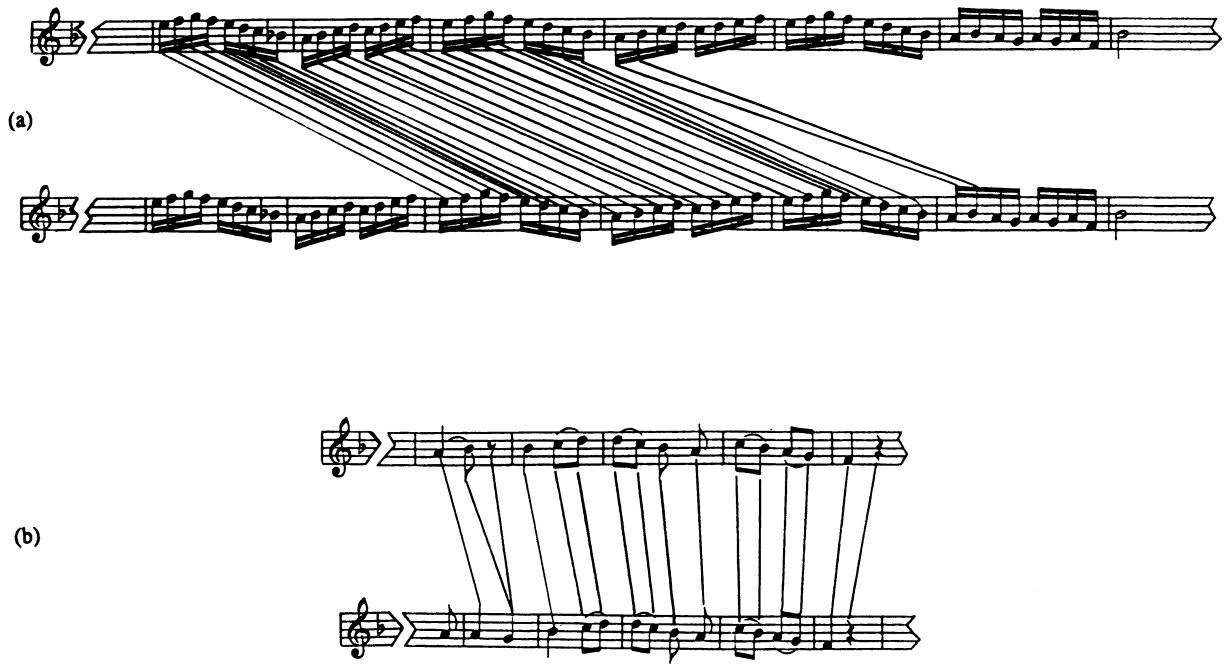


Figure 9. Trace associated with two local alignments of Mozart's Alleluja.

has the advantage of performing a "global" comparison. The criterion of similarity used encompasses both melodic and rhythmic aspects with relative importance governed by a single parameter. It is not affected by gross differences in key, mode or tempo. It also has several disadvantages. A melody in one score which recurs several times in another can only be linked with one of these instances. If two tunes are identical except for the displacement of one portion near the beginning of one and the end of the other, many correspondences will not be identified since the trace lines of the displaced bars cannot cross those of the rest of the tune. Another disadvantage is that the whole scores given as input to the algorithm *have* to be "matched," even the portions having no similar counterpart in the other score. All these disadvantages vanish when the (second) algorithm which finds local similarities, with its flexible length parameter, is applied.

Applications of these algorithms could be extended to any kind of sequences in which time is an essentially quantitative dimension (rhythm, in music) and not simply an ordinal parameter (as in speech recognition where compressions and expansions of an elastic time axis are allowed). Similar studies might be carried out in a framework not restricted to tonal music and generalized to polyphonic scores. Possibilities for application are numerous, for comparative analysis of different versions of folksongs and production of derivation trees in ethnomusicology or in medieval musicology (e.g. the study of Gregorian chant), criticism, as an adjunct to automated composition, for the study of the development of a composer, and even for questions of copyright.

References

Anonymous. *Radio City Album of Soprano Solos.* New York: Edward B. Music Corporation, 1932, pp. 2—7.

Dillon, M. and M. Hunter. "Automated Identification of Melodic Variants in Folk Music." *Computers and the Humanities*, 16 (1982), 107—17.

Duschenes, M. *Méthodes de flûte à bec. Vol. II.* BMI Canada Ltd, Toronto, 1962, pp. 69—72.

Kruskal, J. B. and D. Sankoff. "An Anthology of Algorithms and Concepts for Sequence Comparison." In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Ed. D. Sankoff and J. B. Kruskal. Reading, MA: Addison-Wesley, 1983, pp. 293—96.

Kruskal, J. B. and M. Liberman. "The Symmetric Time-Warping Problem: From Continuous to Discrete." In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Ed. D. Sankoff and J. B. Kruskal. Reading, MA: Addison-Wesley, 1983, pp. 125—59.

Logrippo, L. and B. Stepien. "Cluster Analysis for the Computer-Assisted Statistical Analysis of Melodies." *Computers and the Humanities*, 20 (1986), 19—33.

Mozart, W. A. Ah! vous dirai-je, maman. K300, 1781—82.

Mozart, W. A. Alleluja. Extract of motet "Exultate." K165, 1773.

Rao, C. R. "Use and Interpretation of Principal Components Analysis in Applied Research." In *Sankhya. The Indian Journal of Statistics*. Series A, 26 (1965), 329—58.

Sankoff, D. and J. B. Kruskal, eds. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reading, MA: Addison-Wesley, 1983.

Smith, T. F. and M. S. Waterman. "Identification of Common Molecular Subsequences." *Journal of Molecular Biology*, 147 (1981), 195—97.

Stech, D. A. "A Computer-Assisted Approach to Micro-Analysis of Melodic Lines." *Computers and the Humanities*, 15 (1981), 211—21.

Williams, W. T. and G. N. Lance. "Hierarchical Classificatory Methods." In *Statistical Methods for Digital Computers*. Vol. III of *Mathematical Methods for Digital Computers*. Ed. K. Enslein, A. Ralston, H. S. Wilf. New York: Wiley, 1977, p. 280.