Contour Similarity Algorithms

*Marcos da Silva Sampaio
Universidade Federal da Bahia
marcos@sampaio.me

Abstract: *The Musical Contour literature provides multiple algorithms for melodic contour similarity. However, most of them are limited in use by the melody length of input data. In this paper I review these algorithms, propose two new algorithms, compare them in three experiments with contours from the Bach Chorales, from a Schumann song and automated generated, and present a brief review of the contour and similarity literature.*
Keywords*: Music Contour Theory. Melodic Similarity. Algorithms. Music Analysis. Computational Musicology.*

I. Introduction

Similarity is a paramount concept in Music. It is important because it supports the recognition of music structures such as motives, themes, and chords. The measurement of music similarity is in the heart of the Music Analysis, as the base for identity tests and comparison of musical units. Music similarity has been discussed by many researchers of Music Theory, Ethnomusicology, Music Psychology and Cognition, and Music Information Retrieval fields.

The music contour similarity is one of the three kinds of melodic similarity, together with the pattern and global similarity [8]. There are many algorithms to measure the similarity between melodic contours in the literature, such as the Oscillation spectrum correlation [26, 27], the Embedded contour patterns [17], and the Rigid and Fuzzy comparison [22]. Despite the abundance of contour similarity algorithms, there is none that performs well with both small and large contours. Algorithms such as Rigid and Fuzzy comparison are only able to compare contours with the same size, the Embedded contour algorithm has high computational complexity (factorial), and Oscillation spectrum algorithm doesn't perform well with small contours with less than six points—See [7] for further information concerning computational complexity.

In this paper, I present two new contour similarity algorithms to fill this gap, a comparison with the available contour similarity algorithms and a brief review of the similarity literature.

II. Similarity

Many researchers have addressed concepts and measures concerning melodic similarity [8, 21, 28, 27, 5, 9, 15, 19, 4]. According to Eerola [8], two melodies are considered similar if they contain similar short patterns of pitches or rhythms, or resemblance shape. He classifies the similarity in three types: pattern, contour and global similarity—a combination of various representations of melodies, however, the nature of melodic similarity is not clear neither in the Psychology nor in the Music Information Retrieval research fields [16, p. 1].

*Thanks to MCTI/CNPQ for the research support.

Melodic similarity is a relative concept and it depends on the chosen music properties set used to compare the melodies [13, p. 1]. In a way, this relativeness helps to understand why of so many measures of similarity available in the literature. These measurements are based on multiple approaches such as the string matching of individual notes and geometric representation of melodies. Technically, the basic techniques for measuring the similarity are edit distance, n-grams, correlation and difference coefficients, and hidden Markov models [19].

The music contour similarity has been measured by multiple algorithms, based on the correlation of contour oscillations spectra [26, 27], the patterns embedded in the compared contours, and on the relations among pairwise contour points [17]. These algorithms return real values from zero to one to represent similarity between contour pairs. Other contour similarity measures returns another kind of information, such as complexity order [2, 6], or compare one contour with an abstract average contour of a collection [22]. In Section IV, I review the algorithms that return real values for comparisons of contour pairs.

III. Music Contour basic concepts

The understanding of a minimum set of contour concepts is necessary to follow this paper's premise[1]. A contour is an abstraction of musical parameters. Technically, a music contour is defined as "a set of points in one sequential dimension ordered by any other sequential dimension" [18, p. 283]. The melodic contour, for instance, is the pitch set (abstracted as contour points) ordered in time. A sequential dimension is an attribute dimension where the values can be ordered. Note pitch and note duration, for instance, are sequential dimensions, because the pitches can be ordered from lower to higher and duration from shorter to longer. Thus, despite the focus on the melody in the algorithm's analysis in Section VI, they can be used to compare contours of any musical dimension.

The central aspect of the contour study is the observation of the relationships among its points. The relation of any pair of contour points (CP) is ternary: one CP is lower, equal to, or higher than the other (See the Comparison Function [18], in the Equation (1)). The contour can be analyzed and represented in a linear or combinatorial[2] way. The linear representation—also known as Contour Adjacent Series (CAS) [11]—regards only the relations between adjacent CPs; the combinatorial contour considers relations between adjacent and non-adjacent CPs. The linear representation is a sequence of "+" and "-" signs to represent the relations between adjacent CPs and the combinatorial representation is a sequence of positive integers, where is the CP with the lower value.

$$CMP(a,b) = \begin{cases} - & : \text{if } b < a \\ 0 & : \text{if } b = a \\ + & : \text{if } b > a \end{cases} \tag{1}$$

For instance, let the contours M and N (Figures 1c and 1d), both from the Mozart's *Eine Kleine Nachtmusik* antecedent and consequent melodies (Figures 1a and 1b). Their Contour Adjacent Series are $M <$ - + - + - + + + $>$ and $N <$ + - + - + - - - $>$, and their combinatorial representation are $M <$ 1 0 1 0 1 0 1 2 3 $>$ and $N <$ 3 2 3 2 3 2 1 2 0 $>$.

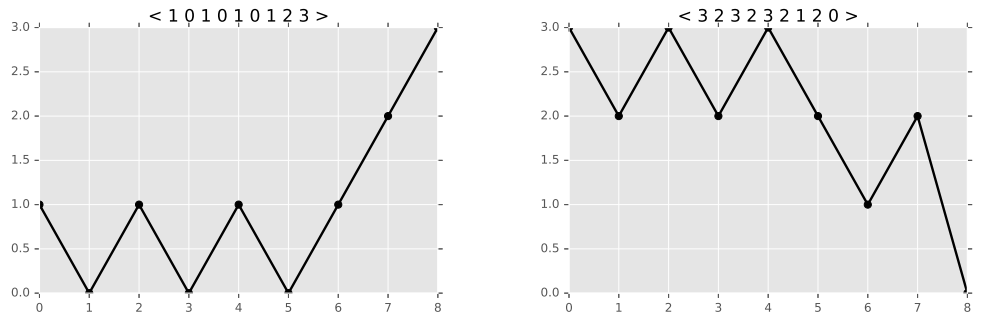
The relations among the contour points are represented in a self-comparison matrix. Two contours are equivalents if they or one of their reflexions—retrograded, inverted, and retrograded-inverted versions—share the same comparison matrix. For instance, the M and N comparison

[1] See [2] and [3] for further information concerning Music Contour Theory.
[2] See [20] and [23] for further information concerning combinatorial contours.

matrices are represented in Tables 1a and 1b. They are not equivalent, once their matrices are different.

(a) *Antecedent melody*

(b) *Consequent melody*



(c) *Antecedent contour*

(d) *Consequent contour*

Figure 1: *Mozart's Eine Kleine Nachtmusik K525 antecedent and consequent.*

	1	0	1	0	1	0	1	2	3
1	0	-	0	-	0	-	0	+	+
0	+	0	+	0	+	0	+	+	+
1	0	-	0	-	0	-	0	+	+
0	+	0	+	0	+	0	+	+	+
1	0	-	0	-	0	-	0	+	+
0	+	0	+	0	+	0	+	+	+
1	0	-	0	-	0	-	0	+	+
2	-	-	-	-	-	-	-	0	+
3	-	-	-	-	-	-	-	-	0

(a) *Antecedent contour matrix*

	3	2	3	2	3	2	1	2	0
3	0	-	0	-	0	-	-	-	-
2	+	0	+	0	+	0	-	0	-
3	0	-	0	-	0	-	-	-	-
2	+	0	+	0	+	0	-	0	-
3	0	-	0	-	0	-	-	-	-
2	+	0	+	0	+	0	-	0	-
1	+	+	+	+	+	+	0	+	-
2	+	0	+	0	+	0	-	0	-
0	+	+	+	+	+	+	+	+	0

(b) *Consequent contour matrix*

Table 1: *Mozart's Eine Kleine Nachtmusik K525 antecedent and consequent contour self-matrices.*

IV. Available contour similarity algorithms

Most of the contour similarity algorithms available in the Contour Theory returns a real number between 0 and 1, other algorithms such as Trace [6] and Multiple Linear Regression [2] returns a polynomial degree to express the contour complexity; and the Fuzzy similarity [22] returns an index value in the comparison of a contour with a set of contours that it belongs. Only the algorithms that return 0 to 1 similarity index between pairs of contours are presented in this paper. For illustration purposes, the melodic contours of Mozart's *Eine Kleine Nachtmusik* antecedent and consequent (Figure 1) are used for similarity calculus with all algorithms. The formalization of these algorithms is available in the Appendix.

	1	0	1	0	1	0	1	2	3
1		-	0	-	0	-	0	+	+
0			+	0	+	0	+	+	+
1				-	0	-	0	+	+
0					+	0	+	+	+
1						-	0	+	+
0							+	+	+
1								+	+
2									+
3									

(a) *Antecedent contour matrix*

	3	2	3	2	3	2	1	2	0
3		-	0	-	0	-	-	-	-
2			+	0	+	0	-	0	-
3				-	0	-	-	-	-
2					+	0	-	0	-
3						-	-	-	-
2							-	0	-
1								+	-
2									-
0									

(b) *Consequent contour matrix*

Table 2: *Comparison between upper triangle of the Mozart's Eine Kleine Nachtmusik K525 antecedent and consequent contour self-matrices.*

i. Rigid Matrix (CSIM)

The Contour Similarity (CSIM) [17] is based on the equivalence of all pairwise contour points of both compared contours. Equal values are summed and divided by the number of relations (See Algorithm 1). This calculus can be made also using the upper right-hand triangle of their comparison matrix.

For instance, the equivalent values of the Mozart's contours M and N matrices (Figure 1) are represented in gray color in Table 2. The similarity between these matrices is 0.44 (16/36). However, the similarity value must be the highest of the comparison among all the reflexions of the two given contours (the original, inverted, retrograded and retrograded-inverted versions). The similarity values for these forms of M and N contours are 0.33, 0.44, 0.5 and 0.55. Thus, the algorithm returns 0.55 as the similarity between the contours M and N. This similarity value is obtained by more than one combination of these reflexions versions, such as between the contour $M < 1\ 0\ 1\ 0\ 1\ 0\ 1\ 2\ 3 >$ and $I(N) < 0\ 1\ 0\ 1\ 0\ 1\ 2\ 1\ 3 >$.

I prefer to call this algorithm as Rigid Similarity Algorithm to differ to the Fuzzy one. In Section VI this algorithm is abbreviated as CMS (Contour Matrix Similarity).

ii. Embedded contours (ACMEMB)

The All Mutually Embedded Contours similarity index (ACMEMB) is presented by [17] as a solution to compare contours with different sizes. It is a type of global/local pattern similarity algorithm. All the contour subsequences embedded in both given contours are calculated and the number of subsequences embedded in both given contours is divided by the total number of subsequences.

The calculus of this index demands the subroutines *TRANSLATION* (Algorithm 2), *CEMB* (Algorithm 3), *ALLCEMB* (Algorithm 6), and *COUNT* (Algorithm 5). The *TRANSLATION* algorithm returns a normalized version of the contour. For instance, a subsequence $< 3\ 6\ 1 >$ is normalized to $< 1\ 2\ 0 >$. The *CEMB* algorithm is a modified Combination algorithm[3]. It returns all the given contour m-sized subsequences combinations. The only change to the combination algorithm is the contour translation of the subsequences. For instance, the function $CEMB(3, A)$

[3]See https://rosettacode.org/wiki/Combinations for multiple implementations of combinations algorithms in multiple languages.

returns the contour 3-sized subsequences $< 0\ 2\ 1 >$, $< 0\ 1\ 2 >$, $< 0\ 1\ 2 >$, and $< 1\ 0\ 2 >$, for the given contour $A < 0\ 2\ 1\ 3 >$.

The *ALLCEMB* algorithm returns all the possible subsequence combinations of a given contour, with sizes 2 to n, where n is the contour size. These subsequences are used in the *ACMEMB* (Algorithm 6) to return the similarity value between the given contours. For instance, the Embedded contour similarity value for the Mozart's contours M and N (Figure 1) is 0.28.

In Section VI this algorithm is abbreviated as EMB.

iii. Contour correspondence

Ellie Hisama [12] proposed the contour deviance measure, the number of deviations between the adjacency series of two given contours. In order to return a similarity index, this operation could be inverted and return the correspondences between the adjacency series of the two given contours. This value could be divided by the series size, to return a real number between 0 and 1, such as in the algorithm 7. Thus, the similarity index between the Mozart's contours M and N is 0.75.

In Section VI this algorithm is abbreviated as COR.

iv. Correlation of contour oscillation spectrum amplitude (OSC)

In the contour oscillation spectrum approach, the contour profile is considered as a sample of a complex wave. The similarity between the given contours is the correlation value between the amplitude spectra of both contours. This calculus can be divided into two parts: an adapted inverse (Fast) Fourier Transform to obtain the wave partials amplitudes[4] (Algorithm 8), and a Pearson's correlation between these amplitudes.

For instance, the oscillation spectra of the Mozart's contours M and N are $[(0.46 - 0.97j), (0.36 - 1.89j), (0.19 - 2.62j), (0.25 - 1.5j)]$ and $[(0.4 + 1.8j), (0.14 + 0.69j), (0.19 + 1.57j), (0.48 - 0.02j)]$, respectively. The amplitude is given by the real part of these complex numbers, in polar coordinates. The correlation between the amplitude of these spectra is 0.14, the similarity value between their respective contours.

In cases where the contour spectra have different sizes, the last partials of the bigger contour are discharged. For instance, let the contours $A < 1\ 0\ 3\ 2\ 1 >$ and $B < 2\ 0\ 2\ 1\ 3\ 2\ 4 >$, and their spectra $[(0.55 - 3.00j), (0.46 - 0.72j)]$ and $[(0.57 - 1.57j), (0.33 - 1.57j), (0.52 - 1.57j)]$, the last partial of B—$(0.52 - 1.57j)$—is not considered for the correlation. In this case, the correlation is 0.19.

I am using the first version of the algorithm, proposed in 1999, with unweighted contours, to follow Friedmann [11] initial definitions in a more strict way: the repeated adjacent pitches are not taken into account and the real intervals between pitches are discarded. Thus, I don't use time weighted pitches, as proposed by Schmuckler [27].

In this paper, we use the acronym OSC to refer to the Correlation of contour oscillation spectrum amplitude.

V. Proposed algorithms

I propose two new contour similarity algorithms: the Adjacent Global Pattern Contour Similarity Algorithm (AGP) and the Adjacent Edit Distance Contour Similarity Algorithm (AED). Both

[4] See https://rosettacode.org/wiki/Fast_Fourier_transform for multiple implementations of the Fast Fourier Transform (FFT) algorithm. In the Algorithm 8, I present an Inverted Discrete Fourier Transform, instead of FFT, just for a better comprehension.

contours are based on established string similarity algorithms.

i. Adjacent Global Pattern Similarity Algorithm (AGP)

The Adjacent Global Pattern Contour Similarity Algorithm (AGP) is an application of the standard Ratcliff/Obershelp Pattern Recognition algorithm [24] to handle the contour similarity problem. This algorithm is available in the Dictionary of Algorithms and Data Structures [1] and is implemented in Difflib Python library[5] [10].

Originally, the Ratcliff/Obershelp algorithm returns an index for the similarity between two given strings. The algorithm finds the longest common substrings of the both given strings, counts the remaining characters and divides this number by the sum of the two strings sizes.

For instance, let the strings "Pennsylvania" and "Pencilvaneya" (author's example), the substrings "Pen", "lvan" and "a" are common to both strings. The remaining characters are "nsy" and "ci", "i" and "ey". There are 8 remaining characters and the strings have 12 characters, each. Thus, the similarity index of the strings "Pennsylvania" and "Pencilvaneya" is 0.66 ($\frac{(12+12)-8}{12+12}$).

Our proposition is to use this algorithm to obtain similarity index between two given contour linear representations strings. For instance, the linear representations of the Mozart's contours M and N are $<$ - + - + - + + + $>$ and $<$ + - + - + - - - $>$, expressed as the strings "-+-+-+++" and "+-+-+—", respectively. The AGP similarity value is 0.75.

This algorithm favors both local and global contour features. In the global level, the longest common sequences are previously aligned, keeping possible large-scale contour structure. In the local level, the adjacent relations are compared and only the differences between common sequences decrease the similarity.

ii. Adjacent Edit Distance (AED)

The Adjacent Edit Distance Contour Similarity Algorithm (AED) is an application of the standard Levenshtein Edit Distance algorithm [14] to handle the contour similarity problem. This algorithm is available in the Dictionary of Algorithms and Data Structures [1] and is implemented in many programming languages[6].

Originally, the Levenshtein algorithm returns an integer as the distance value between two given strings. The distance is given by the number of insertions, deletions or substitutions to transform one string into the other. For instance, let the strings m "Pencilvaneya" and n "Pennsylvania", there are three substitutions ("c" with "n" at position 4, "i" with "s" at position 5 and "e" with "i" at position 10), one insertion ("y" at position 5) and one deletion ("y" at position 11) to transform m into n. Thus, the distance between m and n is 5.

My proposition is to use the Levenshtein algorithm as the base to obtain the similarity index between two given contour linear representations strings. To calculate the similarity index, the ratio between the distance and the size of the bigger string is subtracted from 1 (Equation 2).

$$AED(m,n) = 1 - \frac{LEV(m,n)}{MAX(m.length, n.length)} \tag{2}$$

For instance, the linear representations of the contours M and N are $<$ - + - + - + + + $>$ and $<$ + - + - + - - - $>$, expressed as the strings "-+-+-+++" and "+-+-+—". The distance between these strings is divided by the size of the bigger contour. Thus, the AED similarity value is 0.5 (4/8).

[5]The Difflib source code is available at https://github.com/python/cpython/blob/master/Lib/difflib.py.

[6]See https://rosettacode.org/wiki/Levenshtein_distance for implementations of the Levenshtein Distance algorithm.

	OSC	AGP	AED	CMS	EMB	COR
Mean	0.83	0.71	0.59	0.63	0.58	0.50
Std	0.29	0.15	0.20	0.13	0.11	0.27

Table 3: *Statistical summary of data of comparisons among contours of size from 2 to 6 of the various algorithms.*

VI. Algorithms comparison

I used three contour collections to test the algorithms. The first is a sample of contours with sizes from two to six generated automatically. This sample was obtained with confidence level 95% and confidence interval 5%. The second is a sample of contours from the Bach Chorales collection, and the third is composed by the contours of the phrases of the Robert Schumann's Op. 15, n. 7 (*Träumerei*). I run a statistical analysis with the data from the three collections.

I didn't use the CMS, COR and EMB algorithms to analyze Bach's Chorales and Schumann's *Träumerei* Analysis because they are able only for contours with equal size—CMS and COR—or are computational high complex—EMB (See Section VII for further information).

i. Contours with 2 to 6 points generated automatically

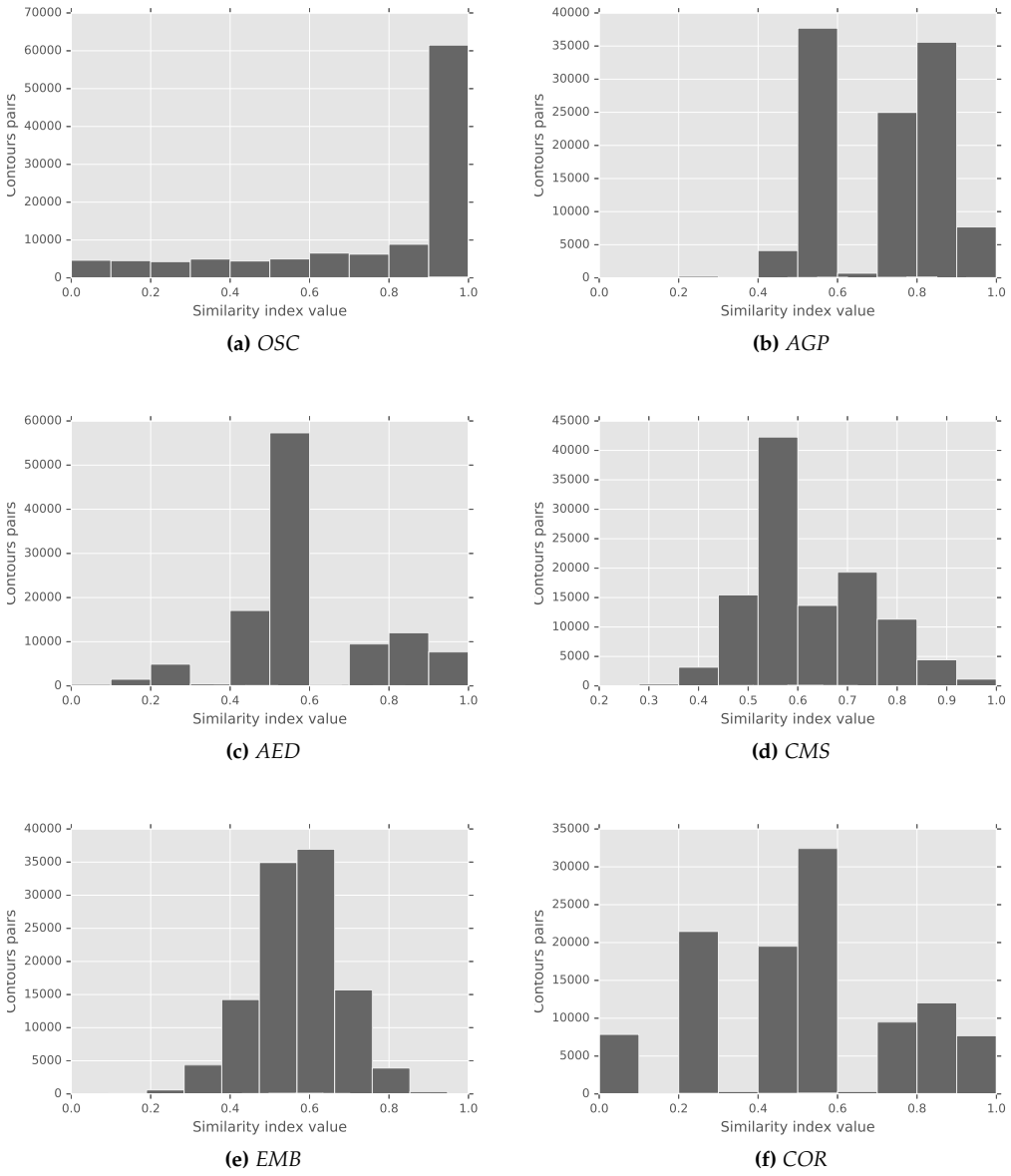
The first experiment dataset is a matrix of the similarity values given by OSC, AGP, AED, CMS, EMB and COR algorithms to 142.845 equal sized contour pairs. These contours—535 in total—are a random sample of contours with size between 2 and 6. The similarity values of each algorithm differ from each other. The data mean ranges from 0.50 to 0.83, and the standard deviation, from 0.11 to 0.29 (See the Table 3). In general, these values are irregularly distributed, except by the EMB algorithm data, normally distributed (See the Figure 2). There is a huge concentration of values in the range between 0.9 and 1.0 in the OSC algorithm data (See the Figure 2a). The causes for this concentration will be discussed in Section VII.

In general, there is a low correlation among these algorithms data (0.21 on average). The highest correlation, 0.86, occurs between AED and COR algorithms data—both algorithms are based on differences in the CAS elements—, and the lowest, 0, occurs between OSC and COR algorithms (See Table 4). The relation among these algorithms data can be viewed in Figure 3. The distribution that involves CMS, AGP and AED have aligned points for specific values—For instance, in Figure 3d, there is a horizontal line in the AGP similarity 0.6 and 0.8. It shows that these algorithms return a small set of values with this collection. This situation doesn't occur in the same way with OSC and OMB algorithms, despite the concentration of OSC similarity value in the 1.0 (See Figure 3b).

The reason of this small set of values is the nature of the algorithm: an integer from zero to the size of the contour divided by the size of the contour. Once the data has operations among equal sized contours, the algorithm returns only a few different values. For instance, contours with four points have CAS with three elements. The results will be only 0/3, 1/3, 2/3 and 3/3.

ii. Bach Chorales phrases analysis

The second experiment dataset is a matrix of the similarity values given by the OSC, AGP and AED algorithms to 70.500 contour pairs. These contours—376 in total—are a random sample of contours from all four voices of the Bach Chorales phrases. This sample has phrases with contours with sizes from one to 28 points, with mean and standard deviation 8.27 and 3.41, respectively.



(a) *OSC* **(b)** *AGP*

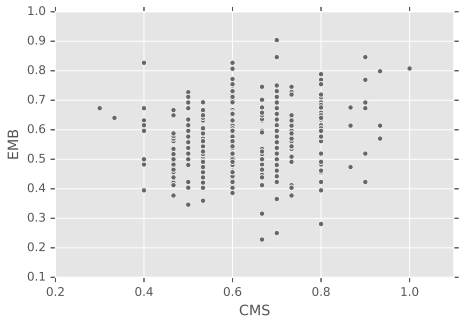
(c) *AED* **(d)** *CMS*

(e) *EMB* **(f)** *COR*

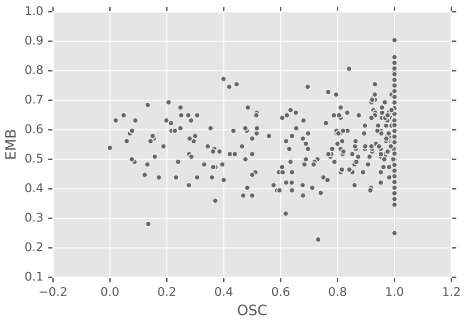
Figure 2: *Algorithm values distribution in comparisons among contours with size from 2 to 6*

	OSC	AGP	AED	CMS	EMB	COR
OSC	1.00	-0.01	-0.04	0.10	0.11	0.00
AGP	-0.01	1.00	0.76	0.13	0.31	0.51
AED	-0.04	0.76	1.00	0.10	0.17	0.86
CMS	0.10	0.13	0.10	1.00	0.16	-0.02
EMB	0.11	0.31	0.17	0.16	1.00	0.12
COR	0.00	0.51	0.86	-0.02	0.12	1.00

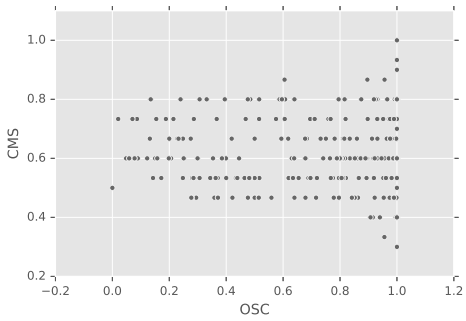
Table 4: *Correlation of algorithm measures of similarity among contours with size from 2 to 6.*



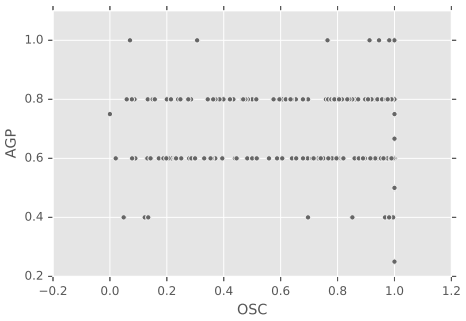
(a) *CMS and EMB*



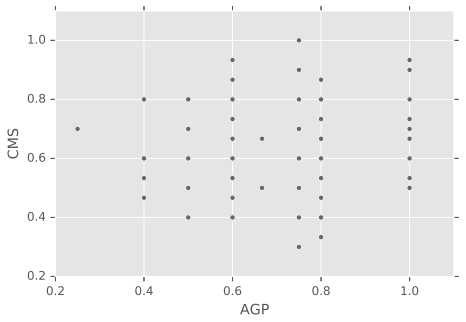
(b) *OSC and EMB*



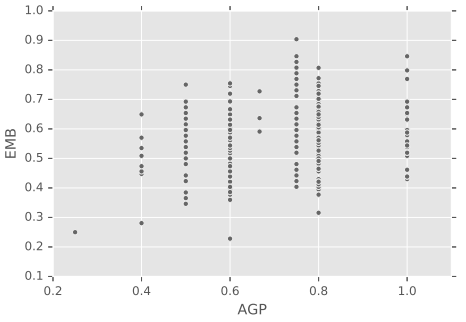
(c) *OSC and CMS*



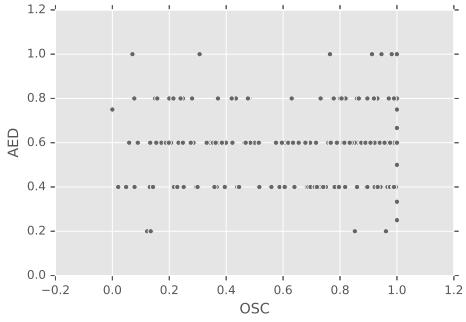
(d) *OSC and AGP*



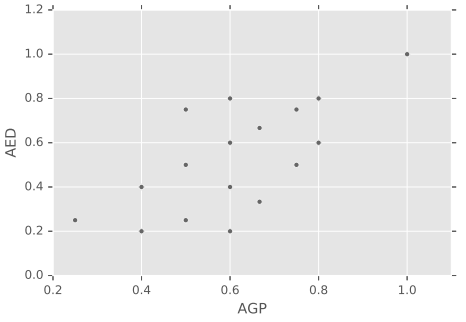
(e) *AGP and CMS*



(f) *AGP and EMB*



(g) *OSC and AED*



(h) *AGP and AED*

Figure 3: *Comparison of algorithms values in contours with size from 2 to 6*

	OSC	AGP	AED
Mean	0.60	0.62	0.52
Std	0.29	0.15	0.14

Table 5: *Statistical summary of data of comparing contours of a random sample of 376 Bach Chorales phrases by the various algorithms.*

	OSC	AGP	AED
OSC	1.00	0.03	-0.01
AGP	0.03	1.00	0.87
AED	-0.01	0.87	1.00

Table 6: *Correlation among algorithms datasets in Bach Chorales*

The similarity values of each algorithm differ from each other, but not as much as in the previous experiment. The data mean ranges from 0.52 to 0.62 and the standard deviation, from 0.14 to 0.29 (See Table 5). These values are normally distributed (AGP and AED algorithms) and fairly uniformly distributed (OSC) (See the Figure 4). There is an expressive concentration of values in the range between 0.9 and 1.0 in the OSC algorithm data (Figure 4a), slightly lower in comparison to the first experiment (Figure 2a).

Table 6 contains the correlation values between the algorithms datasets. Figure 5 contains similarity comparison for pairs of algorithms. There is no correlation between OSC dataset and AGP and AED datasets—with correlation values are 0.03 and -0.01, respectively. The data figures 5a and 5b confirm this lack of correlation. However, there is a strong correlation between AGP and AED datasets (0.87), confirmed by the data in the figure 5c.

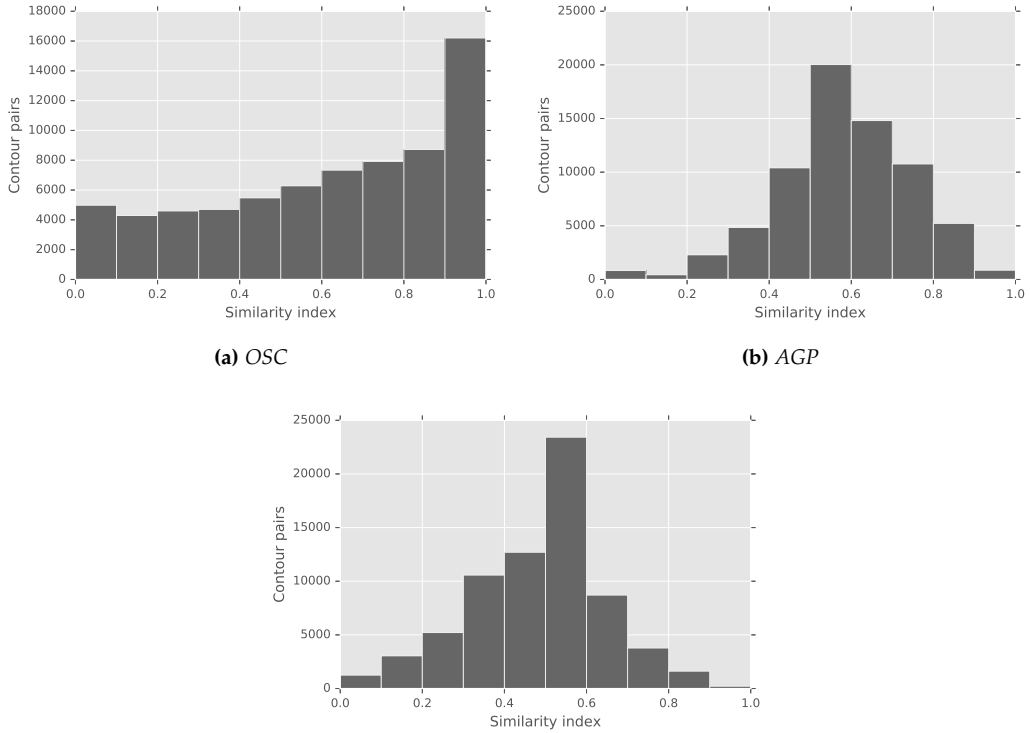
iii. Analysis of Schumann's Op. 15, n. 7—Träumerei

The third experiment dataset is a matrix of the similarity values given by OSC, AGP and AED to 15 contour pairs. These contours—6 in total—were obtained from Schumann's *Träumerei* phrases (See the melody in Figure 6 and the contours in Table 7). This piece's six phrases are equal in size, but with a different number of contour points.

All piece phrases begin with anacrusis and similar contour, but end in different ways. The OSC, AGP, and AED similarity values are very close (See Table 8). The values mean is between 0.81 and 0.86, and the standard deviation, between 0.07 and 0.09 (See Table 9). The correlation between AGP and AED similarity values is high, still slightly higher than in the other experiments (0.96), and the correlation among OSC and both AGP and AED similarity values is expressive (0.53 and 0.50, respectively), much higher than in the other experiments, where there was no correlation

Phrase	Contour
1	< 0 2 1 2 4 6 9 8 7 6 9 3 4 5 7 2 3 4 6 3 >
2	< 0 2 1 2 3 4 10 9 8 7 8 10 5 8 7 6 5 7 4 >
3	< 0 3 2 3 5 7 9 8 7 6 8 4 5 6 5 4 1 >
4	< 0 3 2 3 4 6 9 8 7 6 8 4 5 6 5 4 2 1 >
5	< 0 2 1 2 4 6 9 8 7 6 9 3 4 5 7 2 3 4 6 3 >
6	< 0 3 2 3 5 7 11 10 9 8 7 9 4 5 6 8 4 5 6 8 1 2 3 >

Table 7: *Robert Schumann's Träumerei contours*

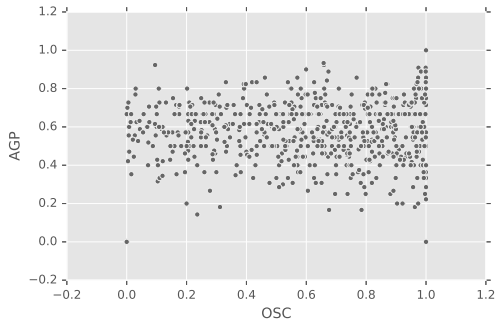


(a) *OSC*

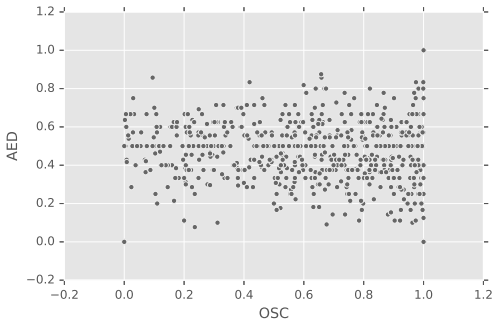
(b) *AGP*

(c) *AED*

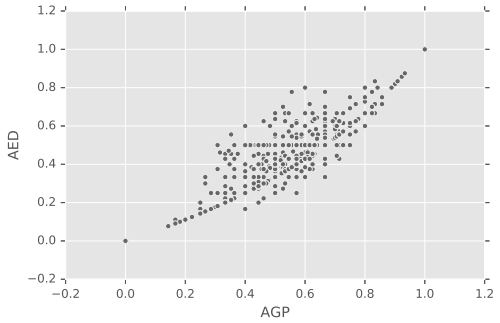
Figure 4: *Contour similarity distributions of a random sample of 376 Bach Chorales phrases.*



(a) *OSC and AGP*



(b) *OSC and AED*



(c) *AGP and AED*

Figure 5: *Comparison similarity values for a random sample of 376 Bach chorales contours.*

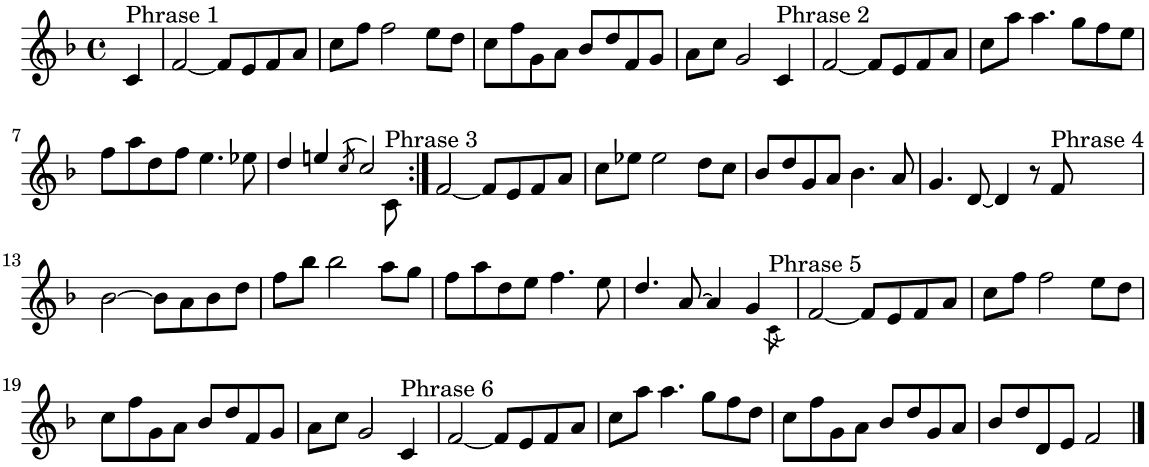


Figure 6: *Robert Schumann's Träumerei melody (Op. 15, n. 7).*

	1	2	3	4	5	6
1	1.00	0.81	0.70	0.74	1.00	0.85
2	0.81	1.00	0.92	0.97	0.81	0.87
3	0.70	0.92	1.00	0.95	0.70	0.86
4	0.74	0.97	0.95	1.00	0.74	0.83
5	1.00	0.81	0.70	0.74	1.00	0.85
6	0.85	0.87	0.86	0.83	0.85	1.00

(a) *OSC*

	1	2	3	4	5	6
1	1.00	0.81	0.86	0.83	1.00	0.93
2	0.81	1.00	0.88	0.91	0.81	0.75
3	0.86	0.88	1.00	0.97	0.86	0.79
4	0.83	0.91	0.97	1.00	0.83	0.77
5	1.00	0.81	0.86	0.83	1.00	0.93
6	0.93	0.75	0.79	0.77	0.93	1.00

(b) *AGP*

	1	2	3	4	5	6
1	1.00	0.79	0.79	0.79	1.00	0.86
2	0.79	1.00	0.83	0.83	0.79	0.68
3	0.79	0.83	1.00	0.94	0.79	0.68
4	0.79	0.83	0.94	1.00	0.79	0.68
5	1.00	0.79	0.79	0.79	1.00	0.86
6	0.86	0.68	0.68	0.68	0.86	1.00

(c) *AED*

Table 8: *Contour similarity values among Schumann's Träumerei phrases. The columns and rows refer to the phrase numbers.*

	OSC	AGP	AED
Mean	0.84	0.86	0.81
Std	0.09	0.07	0.09

Table 9: *Statistical summary of data of comparing contours of a Schumann's Träumerei phrases by the various algorithms.*

(See Table 10).

Finally, the similarity values for the adjacent contours perform a similar shape with the data of the three algorithms (See Figure 7). There is an ascendent similarity until the phrase 4, the lowest similarity between the phrases 4 and 5, and a return to a median similarity value between the phrases 5 and 6. This observation of the similarity progress between adjacent phrases helps to understand how approximation and withdraws—and consequently contrast and continuity—occur throughout a music piece.

VII. Discussion

As seen in Section IV, the Embedded Contour Similarity (EMB) and Oscillation Correlation Similarity (OSC) are the only algorithms provided by the contour literature that are able to

	OSC	AGP	AED
OSC	1.00	0.53	0.50
AGP	0.53	1.00	0.96
AED	0.50	0.96	1.00

Table 10: *Correlation of algorithms measures of similarity among the Schumann's Träumerei phrases contours.*

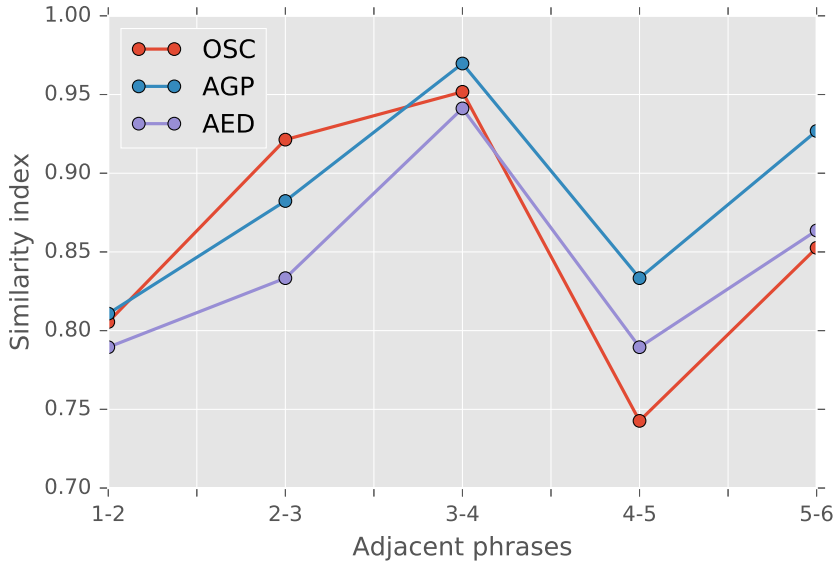


Figure 7: *OSC, AGP and AED similarity values among adjacent phrases in Schumann's Träumerei.*

Algorithm	Complexity	Different sizes
CMS	Quadratic	No
EMB	Factorial	Yes
COR	Linear	No
OSC	Linear-logarithmic	Yes
AGP	Quadratic	Yes
AED	Linear-logarithmic	Yes

Table 11: *Algorithms computational complexity and ability to handle comparisons among contours with different sizes*

compare contours with different sizes. Both proposed algorithms AGP, and AED are also able to compare contours with different sizes.

In terms of computational complexity, the OSC, AGP, and AED have acceptable complexity growth, but EMB not (See the Table 11). The EMB has a high computational complexity, of a factorial order, and is therefore limited to small contours comparisons. The factorial order complexity means that, given a contour with m points, it will take about $m!$ calculations to obtain the similarity value between it and a smaller contour. For this reason, it was not used in the experiments 2 and 3, in Section VI, where there were big contours.

The computational time complexity of the EMB algorithm is factorial in function of the CEMB. The number of combinations of a sequence is given by the binomial coefficient[7] (Equation (3)). Once the number of combinations of a given sequence is increased on a factorial basis, the complexity of the CEMB algorithm will be of a factorial order of computational time complexity. For instance, the number of calculations to obtain the similarity between the contours M and N is

[7]See [25] for further information concerning combinations and their calculus.

in the order of 10^6.

$$C(n,r) = \frac{n!}{r!(n-r)!} \tag{3}$$

Despite this problem of computational complexity, the algorithm returns the most normal curve of similarity values for contours with size from 2 to 6 (See the Figure 2e). It performs better than OSC, AGP and AED algorithms.

Unlike the EMB, the OSC algorithm is able to compare high length contours because has an acceptable computational complexity order, linear logarithmic. It means that given a contour with m points, it will take about $m \log(m)$ calculations to obtain the similarity value between it and a smaller contour.

However, this algorithm has some weaknesses such as theoretical and practical concerns about the use of Fourier analysis with discrete and short series as input—in contrast to the expected sample of an infinitely periodic signal [26, p. 304]. The small size of the input series will cause "edge effects", decreasing the tool's predictive power. Schmuckler defends the use of the tool arguing that the goal "is not the typical predictive forecasting commonly associated with time-series analyses" and that "Fundamentally, Fourier analysis is simply a mathematical decomposition procedure that is applicable to any numerical series." [26, p. 304], but admits that the applicability of Fourier analysis to melodic contour is an open question.

According to Beard, another weakness is the way Schmuckler conducted his experiments: the subjects are not asked to identify their perception of contour similarity, but to classify contour complexity in a scale. Thus, the similarity is derived indirectly from contour complexity [2, p. 186]. He concludes the similarity is achieved by the subjects' responses, not by Fourier model. However, Schmuckler argues "multidimensional scaling literature suggests that derived similarity data produces scaling metrics comparable to more direct similarity measures" [26, p. 317].

Besides the Schmuckler information about "edge effects" in Fourier analysis, the correlation calculus with datasets with only two points is not very useful, once it returns only the values -1, 0 or 1. The Fourier transform of small contours, with less than six points, results in only two partials. Thus, this correlation results will be an "all or nothing" type.

In this way, the Contour Theory provides an algorithm that can be used only with small contours (EMB) and another that can not be used with small contours (OSC). It's difficult to compare two contours with 5 and 12 points, for instance.

For these reasons I proposed the AGP and AED contour similarity algorithms, that can be used with small and large contours. Once Fast Fourier Transform, Levenshtein and Ratcliff/Obershelp algorithms have a linear-logarithmic order of complexity, both OSC, AGP[8] and AED have a similar computational complexity.

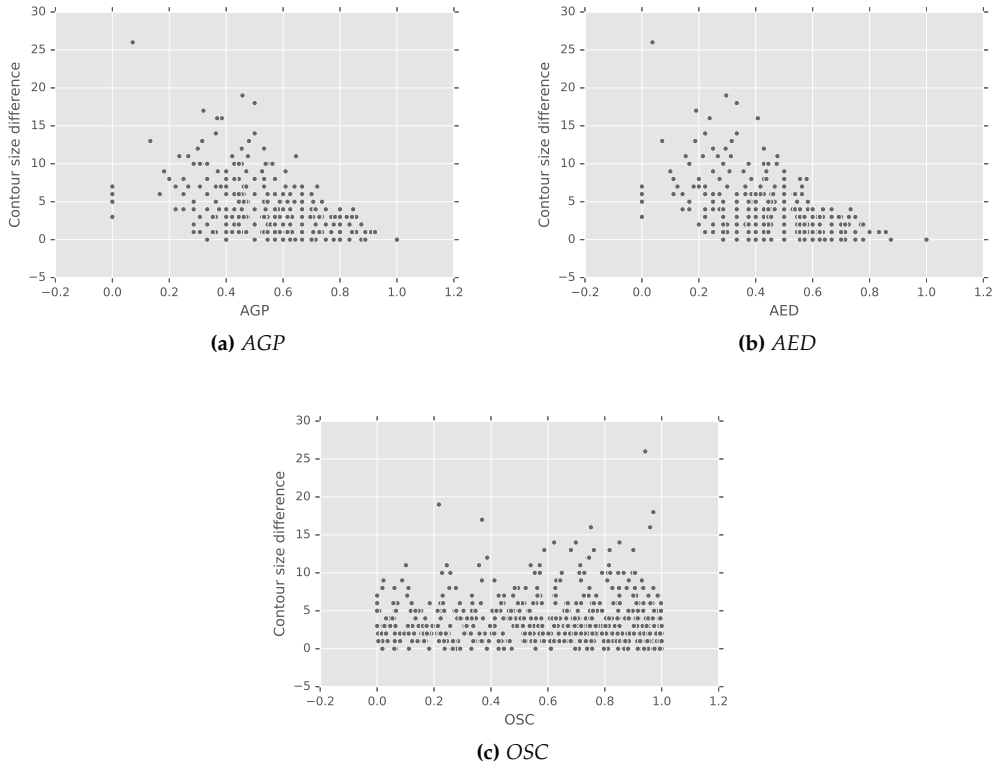
The use of CAS as input ensures the local level similarity features in both AGP and AED algorithms. However, only AGP has a focus on the global aspect, because the longest common strings principle.

The pitfalls of these algorithms are the maximum similarity value for some different contours pairs and the dependency on the contour size difference for the final value. Different contours like $< 0\ 2\ 1\ 3 >$ and $< 2\ 3\ 0\ 1 >$ have AGP similarity index 1, because both are linearly represented as $< +\ -\ +\ >$. And the size difference between contours decreases the similarity values. For instance, contours like $< 0\ 1 >$ and $< 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 >$ have AGP similarity index 0.22, despite the pattern repetition.

[8]The AGP algorithm complexity is cubic in the worst case and quadratic in the expected case, but Difflib's implementation is quadratic in the worst case and linear in the best case [10].

	OSC	AGP	AED
Correlation	-0.08	-0.48	-0.55

Table 12: *Correlation between contour similarity value and contour size difference.*



(a) *AGP*

(b) *AED*

(c) *OSC*

Figure 8: *Relation between contour similarity values and contour size difference*

The size difference between the contour pairs has a negative correlation to the similarity values given by both AGP and AED algorithms—in the Bach Chorales dataset. This correlation is -0.48 and -0.55, respectively. There is no correlation between the size difference and OSC similarity values (See the Table 12 and Figure 8).

Finally, the experiments of the Section VI reveal there are contours with high OSC similarity index values and low AGP ones, and vice-versa. For instance, OSC similarity index between the contours $< 0\ 1\ 2\ 3 >$ and $< 3\ 2\ 1\ 0 >$, $< 0\ 1\ 2\ 3 >$ and $< 4\ 3\ 2\ 1\ 0 >$ is 1.0, and the AGP similarity index, 0.0. In the opposite corner, the contours $< 3\ 2\ 1\ 2\ 3\ 2\ 1\ 0 >$ and $< 4\ 1\ 0\ 2\ 5\ 4\ 3\ 4\ 1 >$ has OSC similarity index 0.01 and AGP similarity index 0.93. This difference occurs in the algorithms pitfals: OSC processing small contours and AGP processing contours with great size difference.

For the pitfalls in the algorithms, I consider the contour similarity measurement as an open problem. I propose AGP and AED to fill the gap of different size contours similarity, but this proposition doesn't close the contour similarity problem.

VIII. Conclusion

In this paper, I presented contour similarity algorithms provided by the literature and introduced two new algorithms. I compared the algorithm's results in three experiments—with generated contours with 2 to 6 points, contours from the Bach Chorales phrases and with contours from a single piece, by Schumann.

The OSC [26] and EMB [17] algorithms are sensitive to the contour sizes. The OSC algorithm is good for contours with more than 5 points, and EMB, for contours with less than 7 or 8 points. Neither is efficient in comparing small with large contours. I introduce two new algorithms to fill this gap, but the difference between contour sizes is a pitfall in both of them. Thus, the contour similarity problem has not yet been solved and demands further research.

References

[1] Vreda Pieterse and Paul E. Black eds. 2017. *Dictionary of Algorithms and Data Structures*. Available at https://www.nist.gov/dads/. Accessed at 20/09/2017.

[2] R. Daniel Beard. 2003. *Contour Modeling by Multiple Linear Regression of the Nineteen Piano Sonatas by Mozart*. Ph.D. dissertation, Florida State University.

[3] Mustafa Bor. 2009. *Contour Reduction Algorithms: a Theory of Pitch and Duration Hierarchies for Post-tonal Music*. Ph.D. dissertation, University of British Columbia.

[4] Chantal Buteau and Guerino Mazzola. 2000. From Contour Similarity to Motivic Topologies. *Musicae Scientiae*, 4(2): pp.125–150.

[5] Emilios Cambouropoulos. 2009. How Similar is Similar? *Musicae Scientiae*, 13(1 Suppl): pp.7–24.

[6] Sean H. Carson.2004. The Trace, Its Relation to Contour Theory, and an Application to Carter's String Quartet No. 2. *Intégral*, 18(2): pp.113–149.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 3 edition.

[8] Tuomas Eerola. 2014. *Similarity, Melodic*, pages 1003–1006. SAGE.

[9] Tuomas Eerola and Micah Bregman. 2007. Melodic and Contextual Similarity of Folk Song Phrases. *Musicae Scientiae*, 11(1998):pp.211–233.

[10] Python Software Foundation. 2017. Difflib—Helpers for Computing Deltas. In *Python Language Reference, version 3.6*. Python Software Foundation, 2017. Available at https://docs.python.org/3.6/library/difflib.html. Accessed at 20/09/2017.

[11] Michael L. Friedmann. 1985. A Methodology for the Discussion of Contour: Its Application to Schoenberg's Music. *Journal of Music Theory*, 29(2): pp.223–248.

[12] Ellie M. Hisama. 2002. The Politics of Contour in Crawford's "Chinaman, Laundryman". In *Gendering Musical Modernism: The Music of Ruth Crawford, Marion Bauer, and Miriam Gideon*, chapter 4. Cambridge University Press.

[13] Juwan Lee, Seokhwan Jo, and Chang D. Yoo. 2011. Coded Melodic Contour Model. *Music Information Retrieval Evaluation eXchange*, pp. 1–2.

[14] Vladimir Iosifovich Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10: p.707.

[15] Beth Logan, Daniel P W Ellis, and Adam Berenzweig. 2007. Toward Evaluation Techniques for Music Similarity. *Proceedings of the 7th ACM/IEEE-CS*, pp.7–11.

[16] Alan Marsden. 2012. Melodic Similarity: A Re-examination of the MIREX2005 Data. In *12th International Conference on Music Perception and Cognition*, Thessaloniki.

[17] Elizabeth West Marvin and Paul A. Laprade. 1987. Relating Musical Contours: Extensions of a Theory for Contour. *Journal of Music Theory*, 31(2):pp.225–267.

[18] Robert Daniel Morris. 1987. *Composition with Pitch-classes: A theory of Compositional Design*. Yale University Press.

[19] Daniel Müllensiefen and Klaus Frieler. 2004. Measuring Melodic Similarity: Human vs. Algorithmic Judgments. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM04)*, pages 1–6, Graz, Austria.

[20] Larry Polansky. 1987. Morphological Metrics: An Introduction to a Theory of Formal Distances. In *Proceedings of International Computer Music Conference*, Champaign/Urbana.

[21] Jon B. Prince. 2014. Contributions of Pitch Contour, Tonality, Rhythm, and Meter to Melodic Similarity. *Journal of Experimental Psychology, Human Perception, and Performance*, 40(6):pp.2319–37.

[22] Ian Quinn. 1997. Fuzzy Extensions to the Theory of Contour. *Music Theory Spectrum*, 19(2):pp.232–263.

[23] Ian Quinn. 1999. The Combinatorial Model of Pitch Contour. *Music Perception*, 16(4):pp.439–456.

[24] John W Ratcliff and David Metzener. 1988. Pattern Matching: The Gestalt Approach. *Dr. Dobb's Journal*, 13:pp.46–72.

[25] Kenneth H. Rosen. 2007. *Discrete Mathematics and Its Applications*. McGraw-Hill, Boston, 6th edition.

[26] Mark A. Schmuckler. 1999. Testing Models of Melodic Contour Similarity. *Music Perception*, 16(3):pp.295–326.

[27] Mark A. Schmuckler. 2010. Melodic Contour Similarity Using Folk Melodies. *Music Perception*, 28(2):pp.169–194.

[28] Anja Volk and Peter van Kranenburg. 2012. Melodic Similarity among Folk Songs: An Annotation Study on Similarity-based Categorization in Music. *Musicae Scientiae*, 16(0):pp.317–339.

Appendix

Algorithm 1 CSIM algorithm

CSIM(A,B), where A and B are two contours with the same size n.

Let $v = 0$
for i from 0 to $n-1$ **do**
  for j from $i+1$ to $n-1$ **do**
    if $cmp(A_i, A_j) = cmp(B_i, B_j)$ **then**
      $v = v + 1$
    end if
  end for
end for
return $\frac{2v}{n^2-n}$

Algorithm 2 Translation algorithm

TRANSLATION(A), where A is a contour sequence with n elements and H is a hash table with records in $\{k, v\}$ format.

Let H
Let $B = SORT(A)$
LET $T = []$
for i from 0 to $n-1$ **do**
  add $\{B[i], i\}$ to H
end for
for i from 0 to $n-1$ **do**
  add $H[A[i]]$ to T
end for
return T

Algorithm 3 CEMB algorithm

CEMB(m, A), where A is a contour sequence with n elements and m is the subsequence size.

if $m = 0$ **then**
  return $[[]]$
end if
Let $combinations = []$
for i from 0 to $n - 1$ **do**
  $x = A[i]$
  $S = CEMB(m - 1, A[i + 1])$
  for j from 0 to $S.length - 1$ **do**
    add $TRANSLATION([x] + S[j])$ to $combinations$
  end for
end for
return $combinations$

Algorithm 4 ALLCEMB algorithm

ALLCEMB(A), where A is a contour sequence with n elements.

Let $combinations = []$
for i from 2 to n **do**
  add $CEMB(i, A)$ to $combinations$
end for
return $combinations$

Algorithm 5 COUNT algorithm

COUNT(ARR), where ARR is an array of contours and H is a hash table with records in $\{k, v\}$ format.

Let H
for i from 0 to $ARR.length$ **do**
  if $ARR[i] \in H.keys$ **then**
    $H[ARR[i]] = 0$
  end if
  $H[ARR[i]] = H[ARR[i]] + 1$
end for
return H

Algorithm 6 ACMEMB algorithm

ACMEMB(A, B), where A and B are two contour sequences and $MERGE$ is a function that merges two arrays in one and $UNIQUE$ returns only the distinct elements of a given array.

Let $Acomb = ALLCEMB(A)$
Let $Bcomb = ALLCEMB(B)$
Let $ALLcomb = MERGE(Acomb, Bcomb)$
Let $Z = UNIQUE(ALLcomb)$
Let $Acount = COUNT(Acomb)$
Let $Bcount = COUNT(Bcomb)$
Let $v = 0$
for i from 0 to $Z.length - 1$ **do**
  if $Z[i] \in Acomb \wedge Z[i] \in Bcomb$ **then**
    $v = v + Acount[Z[i]]$
    $v = v + Bcount[Z[i]]$
  end if
  add $CEMB(i, A)$ to *combinations*
end for
return $\frac{v}{Allcomb.length}$

Algorithm 7 Correspondence algorithm

CORRSIM(A,B), where A and B are two contours with the same size n.

Let $v = 0$
for i from 0 to $n - 2$ **do**
  if $cmp(A_i, A_{i+1}) = cmp(B_i, B_{i+1})$ **then**
    $v = v + 1$
  end if
end for
return $\frac{v}{n}$

Algorithm 8 Adapted Inverse Discrete Fourier Transform algorithm

AIFFT(A), where A is a sequence of numbers, and *real* returns the real part of a given complex number.

Let $N = A.length$
Let $S = []$
for k from 0 to $N - 1$ **do**
  Let $x = 0$
  for n from 0 to $N - 1$ **do**
    $x = x + e^{-2i\pi kn/N} A[k]$
  end for
  $S[k] = real(x/N)$
end for
return S