Musical Sequence Comparison for Melodic and Rhythmic Similarities

Takashi Kadota† Masahiro Hirao† Akira Ishino† Masayuki Takeda†,‡
Ayumi Shinohara† Fumihiro Matsuo†
† Department of Informatics, Kyushu University 33, Fukuoka 812-8581, Japan
‡ PRESTO, Japan Science and Technology Corporation (JST)
E-mail: { kadota, hirao, ishino, takeda, ayumi, matsuo } @i.kyushu-u.ac.jp

Abstract

We address the problem of musical sequence comparison for melodic similarity. Starting with a very simple similarity measure, we improve it step-by-step to finally obtain an acceptable measure. While the measure is still simple and has only two tuning parameters, it is better than that proposed by Mongeau and Sankoff (1990) in the sense that it can distinguish variations on a particular theme from a mixed collection of variations on multiple themes by Mozart, more successfully than the Mongeau-Sankoff measure. We also present a measure for quantifying rhythmic similarity and evaluate its performance on popular Japanese songs.

1 Introduction

Sequence comparison is an important technique in a large variety of applications, e.g., text editing, document retrieval, biocomputing, linguistic analysis, and literary analysis. It will also play a central role in music retrieval or musicological analysis because music can be regarded as a sequence of symbols. A number of studies have been undertaken in such analysis, mainly at the end of the twentieth century (e.g., [19, 6, 5, 7, 8, 9, 15, 16]), and several music information retrieval systems have been proposed (e.g., [18, 14]). In some cases, music is represented as a note sequence, where each note is an ordered pair of pitch and duration, and in some cases the pitches are replaced with so-called *intervals*, the pitch differences between two consecutive notes. Even when dealing with acoustic input, it is often converted into a sequence called a *contour*, i.e., a string consisting of three symbols indicating possible directions of the intervals: upwards, downwards, and a repeat. The reader is referred to [11] for a summary of current practice.

The key question when comparing musical sequences is how to choose, or design, an appropriate measure to quantify their affinities. Mongeau and Sankoff [19] adapted the weighted edit distance measure (see, e.g., [10]), and showed how their measure could be tuned and the framework extended in order to handle musical entities in a meaningful way. They introduced new edit operations, called *consolidation* and *fragmentation*, which associate one note with multiple consecutive notes or vice versa. They reported in [19] that the measure is able to arrange Mozart's nine variations on the theme "Twinkle, Twinkle Little Star" in a reasonable order, which almost coincides with the subjective impression. Their method, however, suffers from the need to tune the weight matrix for the substitution operation. This is done on the basis of a knowledge of harmonics, and the determination of detailed values requires considerable effort.

Although the scheme of weighted edit distance has traditionally been used to quantify affinities between strings, there may be alternative frameworks for designing string similarity measures. As a trial in this direction, Takeda et al. [21] introduced a formal framework named *string resemblance systems* (SRSs). In their framework, the similarity of two strings can be viewed as the maximum score of a pattern that matches them both. The differences among the measures are therefore the choices of the (1) *pattern set* to which common patterns belong, and (2) *pattern score function*, which assigns a score to each pattern.

For example, if we choose the set of *patterns with variable length don't cares* and define the score of a pattern to be the number of symbols in it, then the obtained measure is the length of the longest common subsequence (LCS) of two strings. In fact, the strings `acdeba` and `abdac` have a common pattern $\mathtt{a}\star\mathtt{d}\star\mathtt{a}\star$ which contains three symbols. With this framework, the design and modification of measures is easy. Takeda et al. [21] designed some measures to quantify affinities among classical Japanese poems (Waka) within this framework, and reported their success in discovering previously unnoticed instances of Honkadori (poetic allusion), one important rhetorical device in Waka poems based on specific allusion to earlier famous poems.

In this paper, we use the framework of SRSs and endeavor to design good similarity measures for musical sequence comparison. Musical similarities we wish to measure are not gross similarities in overall key signature, tempo, or mode between two scores. We are interested in local similarities in *melody* and *rhythm*. For developing effective measures to quantify the two kinds of similarities, we employed a stepwise-improvement approach. That is, we started with a very simple measure, and then improved it step-by-step by analyzing its weaknesses. The measure we finally obtained is still simple, but better than that of Mongeau and Sankoff in the sense that only two parameters need to be determined and that it can distinguish variations on a particular theme from a mixed collection of variations on multiple themes by Mozart, whereas the Mongeau-Sankoff measure has difficulty with this task.

It should be emphasized that the aim of the present paper is *not* to establish a reliable measure for musical comparison. It is simplistic to consider that there is a unique best measure. There can be various types of affinities of musical entities, and therefore "goodness" of measure varies depending on the users and their particular interests. Thus, we believe, it is more significant to give a good framework under which we can easily design, or modify, a similarity measure so that it is sensitive to the resemblances we wish to quantify.

2 Dissimilarity measure by Mongeau and Sankoff

This section gives an overview of the dissimilarity measure by Mongeau and Sankoff, and then describes this measure's problems.

2.1 Mongeau and Sankoff's measure

Mongeau and Sankoff [19] defined a dissimilarity measure for musical sequence comparison. They considered a monophonic score as a sequence of ordered pairs of pitch and duration. A rest is a special note whose pitch is a dummy symbol. To make their measure transposition invariant, the pitch of each note is encoded as the relative position from the tonic in semitone-units. We remark that the musical key must be known for such encoding, and determination of the key is often difficult without human intervention. Note durations are coded in 16th note units.

Their dissimilarity measure is a type of weighted edit distance. The basic edit operations are the *substitution*, *insertion* and *deletion* of one note. The weight on a substitution of two notes a and b is a linear combination of two quantities:

$$w(a,b) = w_{interval}(a,b) + k_1 w_{length}(a,b),$$

where k_1 is the relative contribution of length difference versus that of pitch difference (i.e., interval), and is determined empirically. While the length weight $w_{length}(a,b)$ is simply the difference of the lengths of the two notes a and b, the interval weight $w_{interval}(a,b)$ is linked to the consonance of the interval. The insertion and deletion operations are considered as a substitution of two notes, one of which has a duration of zero. In addition to the three edit operations, special edit operations called *consolidation* and *fragmentation* are introduced. These associate one note with multiple notes, and vice versa. The interval weight on a fragmentation is the sum of the interval weights between each of the replacing notes and that of the replaced notes, whereas the length weight is the difference between the total length of the replacing notes and the length of the replaced notes. The interval and the length difference weights are defined in a similar way for a consolidation.

The main disadvantage is that a large number of parameters must be tuned to musical sequences that are to be compared.

2.2 Evaluation of Mongeau and Sankoff's measure

Mongeau and Sankoff showed a dynamic programming-based algorithm for computing the dissimilarity value according to the measure, and then applied it to the set of all possible pairs of Mozart's nine variations on the theme: Ah! vous dirai-je maman (K.265, "Twinkle, Twinkle Little Star"). They reported that the result confirmed subjective impressions of the patterns of similarities among the variations, but only the result on K.265 was presented in [19]. For this reason, we first carried out additional experiments against: K.25, K.354, K.398, K.460, and K.501 by Mozart, six Variations on Paisiello's duet "Nel cor piu", 16 Variations on opus 32 "Eroica", and seven Variations on "God Save the King" by Beethoven, and "Menuet con Variazione" by Anna Bon di Venezia. We confirmed that the result almost coincides with subjective impressions.

Next, we merged the themes and variations of K.265, K.25, and K.354, and applied the algorithm to them. The result is shown in Table 1. In the table, "K.265-th" and "K.265-5" mean the theme and the fifth variation of K.265, respectively. We remark that the arrangement of the data of K.265 differs from that of the data used by Mongeau and Sankoff, and the obtained dissimilarity values do not match theirs for this reason. It is observed that seven of the most similar 10 items in the list for the theme of K.265 are occupied by variations of K.25 and K.354. The situation on the list for the theme of K.25 is similar. This means that some

Table 1. Partial results of dissimilarity measure by Mongeau and Sankoff.

rank	K.265-th		K.25-th	
	title	dissimilarity	title	dissimilarity
1	K.265-5	33.0	K.25-7	12.4
2	K.25-th	36.7	K.25-1	17.6
3	K.25-7	38.3	K.25-5	27.3
4	K.25-5	46.4	K.354-4	36.7
5	K.265-11	46.8	K.265-12	43.3
6	K.25-1	49.1	K.265-5	50.0
7	K.354-th	51.8	K.265-th	50.7
8	K.265-9	53.3	K.25-3	53.8
9	K.354-2	55.5	K.354-7	56.2
10	K.354-6	57.0	K.265-11	56.6

variations on a theme are very similar to another theme, as compared with its own variations. This does not coincide with subjective impressions. When this similarity measure is used, this point will be crucial in any melodic similarity search. Why does the measure make such an assignment of dissimilarity values? One of the main reasons would be that the insertion or deletion of a note can cause a poor correspondence between the notes of two melodies.

3 A unifying framework for string similarity

This section briefly sketches the framework of string resemblance systems according to [21]. In practical applications such as biological sequence comparisons, it is often preferred to measure *similarity* rather than distance between two given strings. We in this paper regard a distance measure as a similarity measure by multiplying the distance values by -1. Also, Gusfield [10] pointed out that in dealing with string similarity the language of alignments is often more convenient than the language of edit operations. Our framework is a generalization of the alignment based scheme and is based on the notion of *common patterns*.

3.1 String resemblance systems

Before describing our scheme, we need to introduce some notation. The set of strings over an alphabet Σ is denoted by Σ^*. The length of a string u is denoted by $|u|$. The string of length 0 is called the *empty string*, and denoted by ε. Let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$. Let us denote by $\boldsymbol{R}$ the set of real numbers.

A *pattern system* is a triple $\langle \Sigma, \Pi, L \rangle$ of a finite alphabet Σ, a set Π of descriptions called *patterns*, and a function L that maps a pattern in Π to a language $L(\pi) \subseteq \Sigma$. A pattern $\pi \in \Pi$ *matches* a string $w \in \Sigma^*$ if w belongs to $L(\pi)$. Also, a pattern π in Π is a *common pattern* of strings w_1 and w_2 in Σ^*, if π matches both of them. Usually, a set Π of patterns is expressed as a set of strings over an alphabet $\Sigma \cup X$, where X is a finite alphabet which is disjoint to Σ.

Definition 1 *A* string resemblance system *(SRS) is a quadruple $\langle \Sigma, \Pi, L, Score \rangle$, where $\langle \Sigma, \Pi, L \rangle$ is a pattern system, and Score is a pattern score function that maps a pattern in Π to a real number. We assume that Score(π) is computable in polynomial-time with respect to the description length of a pattern π.*

An SRS is a pair of a pattern system and a pattern score function. Under an SRS, similarity of strings is quantified as follows:

Definition 2 *The* similarity *between strings x and y with respect to $\langle \Sigma, \Pi, L, Score \rangle$ is defined by*

$$\mathrm{SIM}(x, y) = \max\{Score(\pi) \mid \pi \in \Pi \text{ and } x, y \in L(\pi)\}.$$

When the set $\{Score(\pi) \mid \pi \in \Pi \text{ and } x, y \in L(\pi)\}$ is empty or the maximum does not exist, SIM*(x,y) is undefined.*

The definition above regards the similarity computation as *optimal pattern discovery* [20]. Thus our framework bridges a gap between similarity computation and pattern discovery in this sense. In [21], the class of homomorphic SRSs was shown to cover most of the well-known and well-studied similarity (dissimilarity) measures, including the edit distance, the weighted edit distance, the Hamming distance, the LCS measure.

Definition 3 *A* homomorphic pattern system *is $\langle \Sigma, (\Sigma \cup \Delta)^*, L \rangle$, where:*

1. *Δ is a set of* wildcards *with $\Sigma \cap \Delta = \emptyset$.*
2. *$L : \Pi \to 2^{\Sigma^*}$ is a homomorphism such that $L(c) = \{c\}$ for any $c \in \Sigma$ and $L(\pi_1\pi_2) = L(\pi_1)L(\pi_2)$ for any $\pi_1, \pi_2 \in (\Sigma \cup \Delta)^*$.*
3. *It takes linear time to decide whether or not w belongs to $L(\gamma)$ for any $\gamma \in \Delta$ and any $w \in \Sigma^*$.*

Definition 4 *A pattern score function Score defined on $(\Sigma \cup \Delta)^*$ is* homomorphic, *if Score$(\pi_1\pi_2) =$ Score$(\pi_1) +$ Score(π_2) for any $\pi_1, \pi_2 \in (\Sigma \cup \Delta)^*$. We assume that it takes linear time to compute Score(π) with respect to the description length of a pattern $\pi \in (\Sigma \cup \Delta)^*$.*

Definition 5 *A* homomorphic SRS *is a pair of a homomorphic pattern system $\langle \Sigma, (\Sigma \cup \Delta)^*, L \rangle$, and a homomorphic pattern score function Score : $(\Sigma \cup \Delta)^* \to \boldsymbol{R}$.*

We remark that, when Σ is fixed, a homomorphic SRS is determined by specifying (1) the set Δ of wildcards, (2) the limitation of the mapping L to the domain Δ, and (3) the limitation of the mapping *Score* to the domain $\Sigma \cup \Delta$.

As stated before, the class of homomorphic SRSs covers most of the known similarity (dissimilarity) measures. For example, the edit distance falls into this class. Let $\Delta = \{\psi\}$ where ψ is the wildcard that matches the empty string and any symbol in Σ, namely, $L(\psi) = \Sigma \cup \{\varepsilon\}$. Let $Score(\psi) = -1$ and $Score(c) = 0$ for all $c \in \Sigma$. Then, the similarity measure defined by this homomorphic SRS is the same as the edit distance except that the values are non-positive. Similarly, the Hamming distance can be defined by using the wildcard ϕ that matches any symbol in Σ.

We can define the LCS measure by using the wildcard $\star$ that matches any string in Σ^*. Namely, the homomorphic SRS specified by (1) $\Delta = \{\star\}$, (2) $L(\star) = \Sigma^*$, and (3) $Score(\star) = 0$ and $Score(c) = 1$ for any $c \in \Sigma$ gives the LCS measure. Although another definition is possible for this measure which uses the wildcard ψ with $L(\psi) = \Sigma \cup \{\varepsilon\}$, but the common patterns obtained are much simpler.

The weighted edit distance can also be defined as a homomorphic SRS in which the wildcards $\phi(a|b)$ ($a, b \in \Sigma \cup \{\varepsilon\}$ and $a \neq b$) such that $L(\phi(a|b)) = \{a, b\}$ are introduced, and $Score(\phi(a|b))$ is the weight assigned to an edit operation $a \to b$.

The idea of gap penalty (see, e.g., [10]) can be realized by introducing the wildcards of the form $[w]$ with $L([w]) = \{\varepsilon, w\}$ for all $w \in \Sigma^+$. This is one motivation for us to allow an infinite number of wildcards in Δ.

3.2 Semi-homomorphic SRSs

In the definition of a homomorphic SRS, we restrict the pattern score function to a homomorphism from the monoid $(\Sigma \cup \Delta)^*$ to $\boldsymbol{R}$. The score of a pattern is therefore the total sum of the scores of the characters and wildcards occurring in the pattern. Now, we ease this restriction in order to extend the class of homomorphic SRSs. We here consider a combination of a homomorphic pattern system and a non-homomorphic pattern score function that satisfies that $Score(\pi_1\pi_2) \geq Score(\pi_1) + Score(\pi_2)$, for any $\pi_1, \pi_2 \in (\Sigma \cup \Delta)^*$.

Definition 6 *A pattern score function defined on* $(\Sigma \cup \Delta)^*$ *is* semi-homomorphic *if*

$$Score(\pi) = \max \left\{ \sum_{i=1}^{\ell} g(\pi_i) \;\middle|\; \begin{array}{l} \pi_i \in \mathcal{D} \ (i = 1, \ldots, \ell), \\ \ell \geq 0, \text{ and } \pi = \pi_1 \cdots \pi_\ell \end{array} \right\},$$

where $\mathcal{D}$ is a subset of $(\Sigma \cup \Delta)^+$ with $\mathcal{D}^+ = (\Sigma \cup \Delta)^+$, and g is a function from $\mathcal{D}$ to $\boldsymbol{R}$. For the set $\mathcal{D}$ and the function g, we assume that it takes linear time to decide whether "$w \in L(\pi)$" for any $w \in \Sigma^*$ and any $\pi \in \mathcal{D}$, and that it also takes linear time to compute $g(\pi)$ for any $\pi \in \mathcal{D}$.

Definition 7 *A* semi-homomorphic SRS *is a pair of a homomorphic pattern system* $\langle \Sigma, (\Sigma \cup \Delta)^*, L \rangle$, *and a semi-homomorphic pattern score function Score* : $(\Sigma \cup \Delta)^* \to \boldsymbol{R}$.

The idea behind the definition is to find the 'best' factorization of a pattern in $(\Sigma \cup \Delta)^*$ into a sequence of subpatterns each belonging to $\mathcal{D}$. It was originally motivated in modifying the LCS measure so as to fit to finding similar poem [21]. In fact, one of the similarity measures proposed in [21] is a semi-homomorphic SRS that is a pair of a homomorphic pattern system $\langle \Sigma, (\Sigma \cup \{\star\})^*, L \rangle$ with $L(\star) = \Sigma^*$, and a pattern score function *Score* which is designed to be sensitive to 'continuation' of characters in a pattern, namely, $\mathcal{D} = \Sigma^+ \cup \{\star\}$ and

$$g(\pi) = \begin{cases} f(|\pi|), & \text{if } \pi \in \Sigma^+; \\ 0, & \text{if } \pi = \star, \end{cases}$$

where f satisfies $f(n+m) > f(n) + f(m)$ for any positive integers n, m. The measure was proved to effectively find pairs of similar poems, some of which led to new discoveries in Japanese literary studies (see [21]). Also, Measures II and III defined later in this paper belong to the class of semi-homomorphic SRSs.

Whereas the class of semi-homomorphic SRSs is a proper superset of the class of homomorphic SRSs, the classes of similarity measures defined by them are shown to be identical. Namely, every semi-homomorphic SRS can be converted into an "equivalent" homomorphic SRS. The advantage of using a semi-homomorphic SRS lies only in readability of the patterns obtained as a result of similarity computation. The time and space complexities of similarity computation for a homomorphic (semi-homomorphic) SRS was discussed in [21].

3.3 SRSs with non-homomorphic pattern system

As demonstrated so far, we can handle a variety of string (dis)similarity by changing the pattern system and the pattern score function. The pattern systems appearing in the above examples are, however, restricted to homomorphic ones. Here, we shall mention SRSs with non-homomorphic pattern systems A *fragmentary pattern* (an *order-free pattern*) is a multiset $\{u_1, \ldots, u_k\}$ such that $k > 0$ and $u_1, \ldots, u_k \in \Sigma^+$, and is denoted by $\pi[u_1, \ldots, u_k]$. The language of pattern $\pi[u_1, \ldots, u_k]$ is defined to be the union of the languages $\Sigma^* u_{\sigma(1)} \Sigma^* \cdots \Sigma^* u_{\sigma(k)} \Sigma^*$ over all permutations σ of $\{1, \ldots, k\}$. For example, the language of the pattern $\pi[abc, de]$ is $\Sigma^* abc \Sigma^* de \Sigma^* \cup \Sigma^* de \Sigma^* abc \Sigma^*$. Hori et al.[12] proved that the membership problem for fragmentary patterns is NP-complete and the similarity computation is NP-hard in general. However, the problems are polynomial-time solvable when k is fixed.

The pattern languages, introduced by Angluin [1], is also interesting for our framework. A *pattern* is a string in $\Pi = (\Sigma \cup V)^+$, where V is an infinite set $\{x_1, x_2, \ldots\}$ of variables and $\Sigma \cap V = \emptyset$. For example, $ax_1bx_2x_1$ is a pattern, where $a, b \in \Sigma$. The language of a pattern π is the set of strings obtained by replacing variables in π by non-empty strings. For example, $L(ax_1bx_2x_1) = \{aubvu \mid u, v \in \Sigma^+\}$. The membership problem for the Angluin patterns is NP-complete [1], and the similarity computation is NP-hard in general [22]. However, the problems are solvable in polynomial-time when the number of variables occurring more than once within π is bounded by a fixed number k.

Both the SRSs with fragmentary patterns and with Angluin patterns play central rule in finding similar poems from anthologies of classical Japanese poems (Waka) [21, 22].

4 Similarity between phrases

It is natural to conclude that similarity between two whole musical works is determined on the basis of a comparison of their *phrases*. For this reason, we first endeavor to define a measure to quantify resemblance between phrases.

In this section we aim to develop a similarity measure between phrases within the framework of SRSs. We start with a very simple measure. Then we improve it step-by-step by analyzing its weaknesses. Here we show three similarity measures. The first measure is essentially the same as the Hamming distance and falls into the class of homomorphic SRSs, whereas the second and the third belong to the class of semi-homomorphic SRSs. At the end of this section, we estimate the performance of the three measures against the variation data used in Section 2.

4.1 Measure I

Our criteria in designing a measure are as follows.

1. We divide each note of given note sequences into 16th notes and use the obtained sequences consisting of pitches for comparison.

2. We allow only the substitution operation. Namely, we do not allow either insertion or deletion.

3. We consider only whether or not two pitches are identical. Namely, we do not pay attention to whether two pitches are consonant or dissonant.

Mongeau and Sankoff basically regard a note as an indivisible unit, but they allow fragmentation (resp. consolidation), which divides one note into several notes (resp. consolidates multiple notes into one note). As a generalization of their idea in regard to fragmentation and consolidation, we divide each of the notes into 16th notes. Then the obtained sequence is simply a sequence of pitches. For simplicity, we ignore an octave difference, and therefore the number of possible pitches is 12. Our input string is therefore a string over an alphabet consisting of 13 symbols, which represent pitches and a rest. We denote this alphabet by Σ , and denote the 12 pitches by the symbols

$$C,\ C^\sharp,\ D,\ D^\sharp,\ E,\ F,\ F^\sharp,\ G,\ G^\sharp,\ A,\ A^\sharp,\ B,$$

and the rest by R. We ignore the information as to whether each of the fragmented 16th notes was originally the beginning of a note. Such information will be used when considering rhythmic similarity in Section 6. Note that we cannot measure the similarity between two strings of different length since we allow only substitution.

Now we show a very simple similarity measure, which we refer to as Measure I. This measure falls into the class of homomorphic SRSs. The pattern set is $\Pi = (\Sigma \cup \{\phi\})^*$, where ϕ is a wildcard that matches any symbol in Σ, namely, $L(\phi) = \Sigma$. The pattern score function $\mathit{Score}_1 : \Pi \to \boldsymbol{R}$ is homomorphic and defined by

$$\mathit{Score}_1(c) = 1 \ (c \in \Sigma) \quad \text{and} \quad \mathit{Score}_1(\phi) = 0.$$

That is, $\mathit{Score}_1(\pi)$ is the number of symbols appearing within π. For the sake of maintaining transposition invariance, we define the similarity value between two strings x and y to be the maximum of the 12 possible similarity values obtained by transposing x by a semitone incrementally. Note that this operation does not force us to know the musical key of the input.

4.2 Measure II

Measure I is very simple, but is useful in identifying the repetition of a motif or in detecting very similar phrases in a musical work. However, it may assign a high score to dissimilar phrases, as shown in Fig. 1. Phrases A1 and A2 have a common pattern

$$\pi_{1,2} = C\phi\phi\phi CC\phi\phi GG\phi\phi G\phi G\phi AA\phi AAAA\phi G\phi\phi\phi\phi GG,$$

and its score is 16. On the other hand, Phrases A1 and A3 have a common pattern

$$\pi_{1,3} = \phi\phi CCCCCCGGGGGGGGAA\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi\phi,$$

and its score is also 16. That is, the similarity value between Phrases A1 and A2 is identical to that between Phrases A1 and A3. However, A1 and A2 are quite dissimilar, whereas

Figure 1. Weakness of Measure I. The measure assigns the same value to the pair of A1 and A2 and to the pair of A1 and A3, but the former pair are not very similar while the latter pair are relatively similar.

A1 and A3 are relatively similar. It can be seen that the matches of pitches between A1 and A2 are intermittent and meaningless, whereas those between A1 and A3 are continuous. The subjective impression coincides with this observation. In Measure I we are not concerned with the continuation of matches. We wish to improve the measure so that it assigns a high score only to a long match. A new measure, called Measure II, is a semi-homomorphic SRS, where the pattern system is the same as that of Measure I, and the pattern score function $Score_2$ is defined by $\mathcal{D} = \Sigma^+ \cup \{\phi\}$ and by

$$g(\pi) = \begin{cases} |\pi|, & \text{if } \pi \in \Sigma^+ \text{ and } |\pi| \geq s; \\ 0, & \text{otherwise.} \end{cases}$$

For example, $Score_2(\pi_{1,2})$ is

$$\begin{aligned} &= g(C) + g(\phi) + g(\phi) + \cdots + g(\phi) + g(GG) \\ &= g(C) + g(CC) + g(GG) + g(G) + g(G) \\ &\quad + g(AA) + g(AAAA) + g(G) + g(GG), \end{aligned}$$

which is 12 when $s = 2$ and is 4 when $s = 3, 4$. On the other hand, $Score_2(\pi_{1,3})$ is

$$\begin{aligned} &= g(\phi) + g(\phi) + g(CCCCCCGGGGGGGGAA) \\ &\quad + g(\phi) + \cdots + g(\phi) \\ &= g(CCCCCCGGGGGGGGAA), \end{aligned}$$

which is 16 for any threshold s with $s \leq 16$.

4.3 Measure III

Measure II resolves the weakness of Measure I that we pointed out above with the example of Fig. 1. Measure II, however, has the following weakness. Let us focus on the two phrases in Fig. 2. Phrases B1 and B2 have a common pattern

$$\pi = C\phi\phi CG\phi\phi GA\phi\phi AG\phi\phi GF\phi\phi FE\phi\phi ED\phi\phi DC\phi\phi C.$$

Since the lengths of clusters of symbols in this pattern are at most 2, $Score_2(\pi)$ is 0 whenever the threshold s is greater than 2. The two phrases are, however, quite similar. For this reason, we wish to ignore a short cluster of mismatches. Let t be a threshold for this. The new measure, referred to as Measure III, is a semi-homomorphic SRS, where the pattern score function $Score_3$ is defined by

$$\mathcal{D} = \{\pi \in (\Sigma \cup \{\phi\})^+ \mid \pi \text{ does not contain } \phi^{t+1}\}$$

and

$$g(\pi) = \begin{cases} \text{the number of symbols within } \pi, & \text{if } |\pi| \geq s; \\ 0, & \text{otherwise,} \end{cases}$$

where s, t are thresholds. For $t \geq 2$ and $s \leq 32$, the score of the pattern π mentioned above is $Score_3(\pi) = g(\pi) = 16$, since π itself belongs to $\mathcal{D}$.

4.4 Complexity of similarity computation

For any π_1, π_2 in $\Pi = (\Sigma \cup \{\phi\})^*$, let $\pi_1 \preceq \pi_2$ if and only if

1. $|\pi_1| = |\pi_2| = m$ for some m, and
2. $\pi_1[i] = \pi_2[i]$ or $\pi_1[i] = \phi$ for every i with $1 \leq i \leq m$.

We write this as $\pi_1 \prec \pi_2$ if $\pi_1 \preceq \pi_2$ and $\pi_1 \neq \pi_2$. A common pattern π of two strings $x, y \in \Sigma^*$ is said to be *maximal* if no common pattern π' of them satisfies $\pi \prec \pi'$.

Lemma 1 *For two given strings x, y of equal length, there uniquely exists a maximal common pattern to them, and it can only be computed in linear time and space.*

We can readily show that each of the pattern score functions $Score_1$, $Score_2$, and $Score_3$ of Measures I, II, and III has the following property:

Figure 2. Weakness of Measure II. The measure assigns a low similarity to this pair if the threshold s is greater than 2, even though the two phrases are quite similar.

For any patterns $\pi, \pi' \in \Pi$, $\pi \preceq \pi'$ implies $Score_i(\pi) \leq Score_i(\pi')$.

Therefore, the similarity value of two given strings x, y of equal length is identical to the score of the maximal common pattern of them. We thus have the following result.

Theorem 1 *For each of Measures I, II, and III, the similarity value of two given strings is computed only in linear time and space.*

4.5 Evaluation

We estimated the performances of the three measures by using the variations of K.25, K.265, and K.354. Since there is no established way to extract a phrase from a musical piece, we take all consecutive N-bars as phrases. We tested for N=2, and found the number of phrases obtained was 563. The result is shown in Table 2, where the parameters used in Measures II and III were $s = 12$ and $t = 4$. For example, "K.265-th(21)" indicates the two bars starting at the 21st bar.

First, we compare the result for Measure I with that of Measure II. The points to be emphasized are that the improvement from I to II successfully decreases the similarity values for dissimilar phrase pairs. The pairs having a high similarity value are often a repetition of a motif in a theme, or a highly similar phrase pair between a theme and its variation. However, it should be noted that there may be a risk of decreasing the similarity value of similar pairs. Examining the result for Measure III, by comparing it with that for Measure II, it can be seen that the similarity values of some phrase pairs increase. We confirmed that these phrase pairs are in fact similar. We have thus successfully improved the similarity measure from Measure I to III.

5 Melodic similarity between musical pieces

In the previous section we developed a melodic similarity measure between phrases that are parts of a whole musical piece. Now we try to give a similarity measure between musical pieces based on this measure.

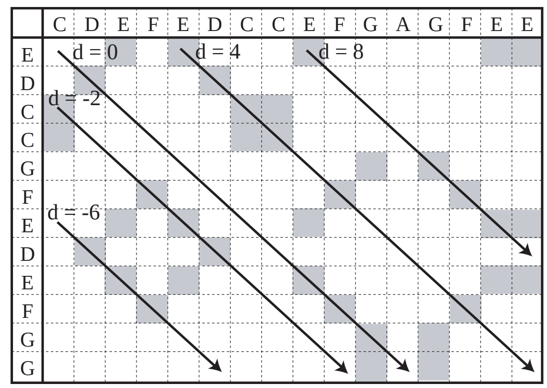

Figure 3. Similarity computation for two whole musical works.

5.1 Definition

For quantifying affinities between two musical pieces x and y, we compare any phrase of x with any phrase of y. Suppose x and y are converted into pitch sequences in 16th note units. Consider a matrix of size $|x| \times |y|$ as shown in Fig. 3. In the matrix, a diagonal d with $-|x| < d < |y|$ corresponds to one alignment. That is, the diagonal d corresponds to a comparison of $x[1 : |x|]$ and $y[1 + d : |y|]$, if $d \geq 0$; and to a comparison of $x[1 - d : |x|]$ and $y[1 : |y|]$, otherwise. Since our measures compare phrases of equal length, the tail of the longer one should be truncated. We compute a similarity value for each diagonal d. The similarity between x and y is defined to be the total sum of the values over all possible diagonals divided by the product $|x| \cdot |y|$. The formal discussion is: Let δ be a similarity measure for strings of equal length in Σ^*. Extend δ so as to compare strings of different lengths by

$$\hat{\delta}(x, y) = \delta(x[1 : \ell], y[1 : \ell]),$$

Table 2. Comparison of three measures. The 15 most similar consecutive two bars against the first consecutive two bars of the theme of K.265 are shown for each of the measures.

rank	Measure I		Measure II		Measure III	
	phrase	similarity	phrase	similarity	phrase	similarity
1	K.265-th(1)	32	K.265-th(1)	32	K.265-th(1)	32
2	K.265-th(21)	31	K.265-th(21)	28	K.265-th(21)	31
3	K.265-th(5)	31	K.265-th(5)	27	K.265-th(5)	31
4	K.265-th(13)	30	K.265-th(13)	20	K.265-th(13)	30
5	K.265-5(5)	23	K.265-th(8)	16	K.265-5(5)	23
6	K.265-th(3)	22	K.265-9(16)	16	K.265-5(13)	21
7	K.265-th(9)	22	K.25-th(16)	13	K.265-th(17)	21
8	K.265-5(13)	21	K.265-9(8)	13	K.265-9(21)	19
9	K.265-th(7)	21			K.265-9(1)	19
10	K.265-th(17)	21	all the others	0	K.265-5(1)	19
11	K.265-9(21)	20			K.265-5(21)	19
12	K.265-9(5)	19			K.265-9(5)	19
13	K.265-9(1)	19			K.265-9(16)	16
14	K.265-5(21)	19			K.265-th(8)	16
15	K.265-5(1)	19			K.265-11(5)	15

where $\ell = \min(|x|, |y|)$. We also use the notation $x[i :]$ for a string x and an integer $i > 0$ to represent the string $x[i : |x|]$. Moreover, we denote by $x[i :]$ the string $x[1 : |x|]$ if $i \leq 0$. Now, let us define the similarity value $\mathrm{SIM}_\delta(x, y)$ between x and y in Σ^+ by

$$\mathrm{SIM}_\delta(x, y) = \frac{1}{|x| \cdot |y|}\Big(\sum_{-|x|<d<|y|} \hat{\delta}(x[1-d :], y[1+d :]\Big).$$

Theorem 2 *If we use Measures I, II, and III as δ, the similarity value $\mathrm{SIM}_\delta(x, y)$ of two strings x, y can be computed in $O(|x||y|)$ time using $O(|x| + |y|)$ space.*

5.2 Evaluation

We estimated the performance of the above-mentioned measure for K.265, K.25 and K.354. We set the parameters of Measure III as $s = 32, t = 5$. The most similar 10 items for each theme are shown in Table 3.

It can be seen that most of the top 10 for each theme are occupied by its variations. This result contrasts markedly with that of the Mongeau-Sankoff measure shown in Table 1. Our measure is thus better than that of Mongeau and Sankoff in identifying similar variations of a theme from a mixture of variations on more than one theme.

6 Rhythmic similarity

We have dealt with melodic similarity, where a melody line is converted into a pitch sequence in 16th note units. Alternatively, let us consider rhythmic similarity in this section. Look at the phrases in Fig. 4. The similarity between Phrases C1 and C2 measured by Measure III presented in Section 4 is relatively small. However, the two phrases sound similar to each other, probably because of their rhythmic similarity.

Table 4. Similar phrases for Song A(18).

rank	melodic similarity		rhythmic similarity	
	phrase	similarity	phrase	similarity
1	Song A(18)	32	Song A(18)	32
2	Song A(55)	32	Song A(55)	32
3	Song A(59)	28	Song A(120)	27
4	Song I(72)	17	Song A(76)	27
5	Song I(20)	16	Song A(39)	27
6	Song I(68)	16	Song A(21)	19
7	Song A(22)	16	Song A(59)	16
8	Song I(24)	16	Song A(22)	16
9	all the others	0	Song A(137)	14
10			Song A(136)	14

We converted the input note sequences into strings consisting of the four symbols: N (beginning of a note), n (middle of a note), R (beginning of a rest), and r (middle of a rest). Then we applied Measure III to 10 randomly selected popular songs. The parameters used are $s = 12$ and $t = 0$ (it is therefore considered as Measure II). The number of distinct consecutive two bars was 1318. The results are shown in Table 4, together with those produced by applying Measure III with $s = 12, t = 4$ to their pitch sequences. Phrases C1 and C2 in Fig. 4 are Song A(18) and Song A(39), respectively. Phrase C2 is not present in the top 10 for melodic similarity, but it ranks third for rhythmic similarity. We have found a number of other phrases that have no melodic similarity, but which have rhythmic similarity to a particular phrase.

Table 3. Most similar 10 items for K.265-th, K.25-th, and K.354-th.

rank	K.265-th		K.25-th		K.354-th	
	title	similarity	title	similarity	title	similarity
1	K.265-5	0.0112	K.25-7	0.0233	K.354-2	0.0077
2	K.265-9	0.0105	K.25-1	0.0120	K.354-6	0.0061
3	K.265-3	0.0030	K.25-6	0.0015	K.354-5	0.0031
4	K.265-1	0.0030	K.25-3	0.0010	K.354-12	0.0024
5	K.265-11	0.0010	K.25-5	0.0010	K.25-3	0.0015
6	K.265-7	0.0005			K.25-6	0.0011
7	K.354-12	0.0003	all the others	0.0	K.354-3	0.0009
8	K.25-3	0.0003			K.265-5	0.0008
9	K.265-12	0.0001			K.354-1	0.0003
10	all the others	0.0			K.354-7	0.0003

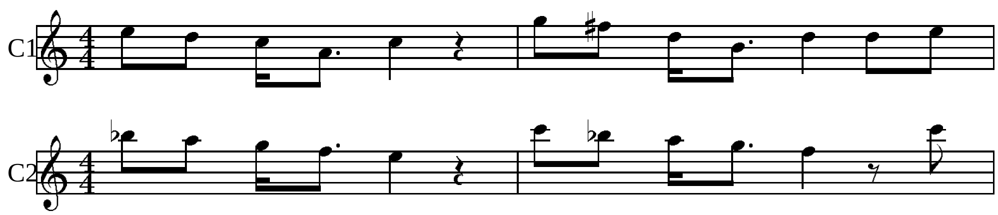

Figure 4. Two phrases similar in rhythm.

7 Conclusion

In this paper, we have addressed the problem of musical sequence comparison for melodic similarity. We started with a very simple measure quantifying melodic similarity, and incrementally refined it within the framework of SRSs. The obtained measure is still simple and has only two parameters to be tuned, but it is superior to the dissimilarity measure proposed by Mongeau and Sankoff. We also presented a measure for quantifying rhythmic similarity and evaluated its performance against Japanese popular songs.

One significant application of our method would be questions of copyright. In Japan, composer Asei Konayashi sued in 1998 another songwriter Katsuhisa Hattori, claiming that Hattori plagiarized his 1966 hit song "Dokomademo Iko" in writing 1992 song "Kinenju." The scores of the two songs are found in Fig. 5, where they are transposed from their own keys to C major, and "Dokomademo Iko" is changed to 4/4 (originally 2/2) for comparison. Although Kobayashi's claim was rejected in 2000 by the Tokyo District Court, there is a considerable amount of similarities between the two songs. Fig. 6 is the maximal common pattern of the two melodies, where we divided each note into 4th notes, not into 16th notes. If we set the tuning parameters as $s = 8$ and $t = 2$, respectively, the similarity value of the two melodies is 98 under Measure III we proposed in this paper. Similar melody search on a large-scale musical database will be helpful for composers who want to prevent their works from being closely similar to some earlier musical pieces. We hope that our studies on musical sequence comparison will give a reliable criterion for this purpose.

Our measures shown in this paper take into account whether two corresponding pitches are identical or not, but they do not relate to the consonance of the two pitches, unlike the Mongeau-Sankoff measure. We wish to realize this idea in our measure by roughly categorizing the relationship of the two pitches, for example into three degrees: consonant, dissonant, and match. Preliminary experiments confirm the effectiveness of the idea [13].

In this paper we have only dealt with monophonic music, but we plan to progress to polyphonic music. In the case of polyphonic music the input is a sequence of sets of notes.

Even when closely similar melody fragments are found, they might not be significant if the fragments are common and frequent. In [21], we proposed a similarity measure where the pattern score function is a function of the *rarity* of the pattern in the database, and empirically proved that the idea effectively excludes worthless affinities. The same idea may be effective when dealing with melodic and rhythmic affinities.

In our method of dealing with melodic similarity, we divide each note of the input note sequence into 16th notes to obtain strings of pitches. The original and the resulting sequences, respectively, can be viewed as a run-length compressed string and its original string, where a note with pitch

Figure 5. The scores of (A) "Kinenju" by Katsuhisa Hattori and of (B) "Dokomademo Iko" by Asei Kobayashi. These songs are transposed from their own keys to C major, and "Dokomademo Iko" is changed to 4/4 (originally 2/2) for comparison.

$$RRCD\ EEE\phi\ \ \phi\phi\phi\phi\ CCCC\ CCCC\ FFFF$$
$$\phi\phi G\phi\ GGGG\ GGGG\ AAAG\ FFGA\ GG\phi\phi$$
$$CCCD\ EEEC\ \phi\phi\phi\phi\ \phi\phi\phi\phi\ \phi\phi CD\ EEE\phi$$
$$\phi\phi\phi\phi\ CCCC\ CCCC\ FFFF\ \phi\phi G\phi\ GGGG$$
$$GGGG\ AAAG\ FFGA\ GG\phi\phi\ CCCD\ EEEC$$
$$\phi\phi\phi\phi\ CCCC\ CC\phi\phi$$

Figure 6. Maximal common pattern of the two melodies, where we divided each note into 4th notes, not into 16th notes.

a and duration of i (in 16th note units) is encoded as an i-times repetition of a symbol a. Hence, this may be efficient if we can perform the similarity computation in run-length compressed strings without expanding them. The problems of computing LCS, Levenstein edit distance, or more complex distance for run-length compressed strings have been studied by several researchers [3, 4, 2, 2, 17]. Efficient similarity computation according to our similarity measures in run-length compressed strings will be a subject for future work.

References

[1] D. Angluin. Finding patterns common to a set of strings. *J. Comput. Sys. Sci.*, 21:46–62, 1980.

[2] A. Apostolico, G. Landau, and S. Skiena. Matching for run-length encoded strings. *J. Complexity*, 15:4–16, 1999.

[3] H. Bunke and J. Csirik. An algorithm for matching run-length coded strings. *Computing*, 50:297–314, 1993.

[4] H. Bunke and J. Csirik. An improved algorithm for computing the edit distance of run-length coded strings. *Information Processing Letters*, 54(2):93–96, 1995.

[5] E. Cambouropoulos, T. Crawford, and C. Iliopoulos. Pattern processing in melodic sequences: Challenges, caveats & prospects. In *Proc. the AISB'99 Convention (Artificial Intelligence and Simulation of Behaviour)*, pages 6–9, 1999.

[6] E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, and Y. J. Pinzon. Algorithms for computing approximate repetitions in musical sequences. In *Proc. 10th Australasian Workshop on Combinatorial Algorithms (AWOCA'99)*, pages 25–27, 1999.

[7] T. Crawford, C. Iliopoulos, and R. Raman. String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11:73–100, 1998.

[8] T. Crawford, C. Iliopoulos, R. Winder, and H. Yu. Approximate musical evolution. *Computers and Humanities*, 2000.

[9] M. Crochemore, C. Iliopoulos, and H. Yu. Algorithms for computing evolutionary chains in molecular and musical sequences. In *Proc. 9th Australian Workshop on Combinatorial Algorithms (AWOCA'98)*, pages 172–184, 1998.

[10] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, 1997.

[11] W. B. Hewlett and E. Selfridge-Field, editors. *Melodic Similarity: Concepts, Procedures and Applications*. MIT Press, 1998.

[12] H. Hori, S. Shimozono, M. Takeda, and A. Shinohara. Fragmentary pattern matching: Complexity, algorithms and applications for analyzing classic literary works. Technical Report DOI-TR-193, Department of Informatics, Kyushu University, May 2001.

[13] T. Kadota, A. Ishino, M. Takeda, and F. Matsuo. On similarity measures for musical sequences. In *Proceedings of the 59th National Convention IPSJ*, volume 2, pages 17–18, 1999. (in Japanese).

[14] K. Lemström and S. Perttu. SEMEX - an efficient music retrieval prototype. In *Proc. First International Symposium on Music Information Retrieval (ISMIR2000)*, 2000.

[15] K. Lemström and J. Tarhio. Searching monophonic patterns within polyphonic sources. In *Content-Based Multimedia Information Access Conference Proceedings (RIAO2000)*, volume 2, pages 1261–1279, 2000.

[16] K. Lemström and E. Ukkonen. Including interval encoding into edit distance based music comparison and retrieval. In *Proc. the AISB'2000 Symposium on Creative & Cultural Aspects and Applications of AI & Cognitive Science*, pages 53–60, 2000.

[17] V. Makinen, G. Navarro, and E. Ukkonen. Approximate matching of run-length compressed strings. In *Proc. 12th Annual Symposium on Combinatorial Pattern Matching (CPM2001)*, 2001. to appear.

[18] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten. The New Zealand digital library MELody inDEX. *D-Lib Magazine*, 1997.

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

[20] S. Shimozono, H. Arimura, and S. Arikawa. Efficient discovery of optimal word-association patterns in large databases. *New Gener. Comput.*, 18(1):49–60, 2000.

[21] M. Takeda, T. Fukuda, I. Nanri, M. Yamasaki, and K. Tamari. Discovering instances of poetic allusion from anthologies of classical Japanese poems. *Theor. Comput. Sci.*, 2001. (to appear).

[22] K. Yamamoto, M. Takeda, A. Shinohara, T. Fukuda, and I. Nanri. Discovering repetitive expressions and affinities from anthologies of classical Japanese poems. In *Proc. 4th International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2001. (to appear).