Extracting performers' behaviors to annotate cases in a CBR system for musical tempo transformations

Josep Lluís Arcos, Maarten Grachten, and Ramon López de Mántaras

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain.
{arcos,maarten,mantaras}@iiia.csic.es, http://www.iiia.csic.es

Abstract. In this paper we describe a method, based on the edit distance, to construct cases of musical performances by annotating them with the musical behavior of the performer. The cases constructed with this knowledge are used in *Tempo-Express*, a CBR system for applying tempo transformations to musical performances, preserving musical expressiveness.

1 Introduction

In the design of a case-based reasoning system, one of the key issues is the case base acquisition. One possible option for acquiring cases is the development of tools for manually incorporating cases into the case base. Another alternative is the development of tools for automatically—or semi-automatically—importing cases from a pre-existing source—either from publicly available archives [4] or problem specific data bases.

For classification or identification tasks, a case can be easily represented by the input data (the case problem) and the class or the enumerated collection of classes the case belongs (the case solution). In other tasks such as planning or design problems, a solution can be a composite structure. Moreover, the solution structure can contain the solution itself and additionally knowledge about the decisions taken when constructing the solution. For instance, *derivational analogy* [5] is based on augmenting the case solution by means of detailed knowledge of the decisions taken while solving the problem, and this recorded information (e.g. decisions, options, justifications) is used to "replay" the solution in the context of the new problem. As originally defined in derivational analogy for planning systems the cases contain *traces* from planning processes performed to solve them; also it is stated that in Prodigy/Analogy stored plans are annotated with plan *rationale* and reuse involves adaptation driven by this rationale [18].

Another task example, where case representation has to be augmented with detailed knowledge of the decisions taken while solving the problem, is the generation of expressive music. As our previous experience in the SaxEx system has demonstrated [1], the problem of the automatic generation of expressive musical performances is that human performers use musical knowledge that is not explicitly noted in musical scores. This knowledge is difficult to verbalize and therefore AI approaches based on declarative knowledge representations have serious limitations. An alternative approach is that of directly using the knowledge implicit in examples from recordings of human performances. Then, when we import recordings to be incorporated as cases in our case base, besides representing the musician performance as the case solution, we also need to *extract* the decisions performed by the musician when playing a melody (in other words, the performers' behavior). These decisions—such as playing additional notes, leaving out some notes, changing note durations, etc—are what we call annotations in this paper.

In this paper we focus our attention on the automatic annotation process developed for *Tempo-Express*, a case-based reasoning system for tempo transformation of musical performances, that preserves expressivity in the context of standard jazz themes. Expressive tempo transformations are a common manual operation in audio post-processing tasks. A CBR system can be a useful tool for automating this task. As we will explain in the next section, the annotation process is used when incorporating cases into the system and when the system must solve a new problem (that is, a new performance must be generated).

Case annotation is based on a dynamic programming algorithm based on the concept of *edit distance*. Edit distance is a technique for assessing the distance between two sequences and calculates this distance as the minimum total cost of transforming one sequence (the source sequence) into the other (the target sequence), given a set of allowed edit operations and a cost function that defines the cost of each edit operation. The output of the algorithm is a quantity that is proportional to the *distance*, or *dis*similarity between the sequences. Moreover, the edit distance provides the sequence of edit operations that yielded this value. Edit distance was first adapted to musical applications by Mongeau and Sankoff in [14].

The paper is organized as follows: In section 2 we briefly introduce the *Tempo-Express* application and their main inference modules. In section 3 we describe the edit distance mechanism. In section 4 we describe the the use of edit distance in the case annotation process. In section 5 we report the experiments performed in annotating cases. The paper ends with a discussion of the results, and the planned future work.

2 Tempo-Express

Tempo-Express is a case-based reasoning system for generating expressive tempo transformations in the context of standard jazz themes. Changing the tempo of a given melody is a problem that cannot be reduced to just applying a uniform transformation to all the notes of a musical piece. When a human performer plays a given melody at different tempos, she does not perform uniform transformations. On the contrary, the relative importance of the notes will determine, for each tempo, the performer's decisions. For instance, if the tempo is very fast, the performer will, among other things, tend to emphasize the most important notes by not playing the less important ones. Alternatively, in the case of slow tempos, the performer tends to delay some notes and anticipate others.

In the development of *Tempo-Express* we are using the experience acquired in developing the SaxEx system [1]. The goal of SaxEx was also to generate expressive music performances but the task was centered on transforming a non expressive input performance into an expressive new sound file taking into account the user preferences regarding the desired expressive output characterized along three affective dimensions (tender-aggressive, sad-joyful, calm-restless). The task of *Tempo-Express* is to perform tempo transformations with 'musical meaning' to an already expressive input performance. That is, a recording has to be replayed at a user required tempo that can be very different from the input tempo.

Below, we briefly present the main *Tempo-Express* modules and the inference flow (see Figure 1). The input of *Tempo-Express* is a recording of a jazz performance at a given tempo T_i (a sound file), its corresponding score (a MIDI file) and the desired output tempo. The score contains the melodic and the harmonic information of the musical piece and is analyzed automatically in terms of the Implication/Realization Model [15] (this analysis is used in the retrieval step and is not further discussed in this paper). The recording is parsed by the performance analysis module that produces a XML file containing the performance melody segmentation (onset points and duration of notes). Then, the melody segmentation is compared to its corresponding score by the Annotation process (see the detailed description in section 4). The annotated performance and the desired output tempo T_o are the input for the case-based reasoning. The task of the case-based reasoning modules is to determine a sequence of operations that achieves the desired tempo transformation while maintaining a form of musical expressivity that is appropriate for that tempo. Finally, the output of the system is a new version of the original melody, at the desired tempo, generated by the synthesis process.

Tempo-Express is implemented in *Noos* [3, 2], an object-centered representation language designed to support the development of knowledge intensive case based reasoning systems. The melodic (performance) Analysis and synthesis processes have been implemented by the Music Technology Group (MTG) of the Pompeu Fabra University using signal spectral modeling techniques (see [17, 8] for a detailed description). Below, we briefly describe the case-based reasoning components.

The Case Base A case is represented as a complex structure embodying three different kinds of knowledge: (1) the representation of the musical score (notes and chords), (2) the musical model of the score (automatically inferred from the score using Narmour's Implication/Realization model and Lerdhal and Jackendoff's Generative Theory of Tonal Music as background musical knowledge [15, 12]), and (3) a collection of annotated performances. For the case acquisition, several saxophone performances were recorded from 5 jazz standards, each one

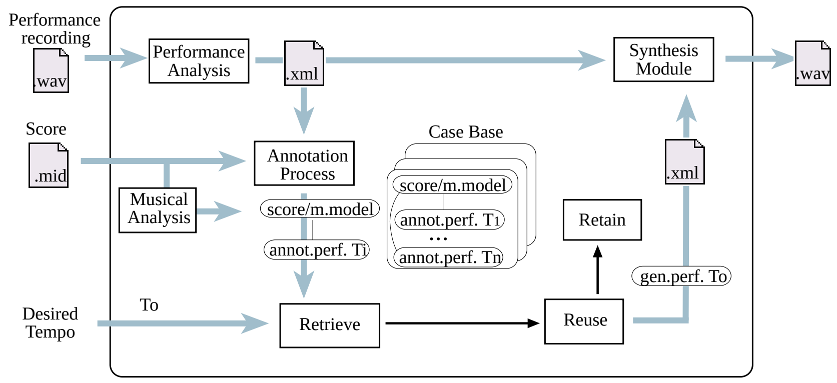

Fig. 1. General view of *Tempo-Express* modules.

consisting of 4–5 distinct phrases. The performances were played by a professional performer, at 9–14 different tempos per phrase. From this, the initial case base was constructed, containing 20 scores of musical phrases, each with about 11 annotated performances (in total more than 5.000 performed notes). See section 5 for more details of the data set.

The Retrieval Step The retrieval mechanism is organized in three phases. In a first phase the input melody is compared with the melodies of the case base using melodic similarity measures (see [9] for a detailed description) for retrieving only those case melodies really similar—For instance, given a slow ballad as input, we are not interested in comparing it with be-bop themes.

In a second phase, we focus on the similarity of the musical models. This phase uses similarity criteria such as: Narmour's musical grouping structures (such as melodic progressions, repetitions, changes in registral directions, etc); the harmonic stability of the note according to jazz harmony; the metrical strengths of the notes; or the hierarchical relations of the notes in the phrase melodies according to the Generative Theory of Tonal Music (GTTM) model.

Finally, in the third phase the similarity among performances are assessed. For example, in figure 1 the annotated input performance (T_i) is compared to the annotated performances ($T_1 \dots T_n$) of the case base whose tempos are the closest to the tempo of T_i. This set of annotated performances is ranked according to the degrees of similarity between their annotations and the annotation of T_i. The output of the retrieval step is a collection of candidate annotations for each note in the input melody.

The Adaptation Step The adaptation mechanism has been implemented using *constructive adaptation* [16], a generative technique for reuse in CBR systems. The adaptation mechanism deals with two kinds of criteria: local criteria and coherence criteria. Local criteria deal with the transformations to be performed to each note—i.e. how retrieved candidate annotations can be reused in each input note. Coherence criteria try to balance smoothness and hardness. Smoothness and hardness are basically contradictory: the first tends to iron out strong deviations with respect to the score, while the second tends to favor strong deviations. The resulting expressive performance is a compromise between the smoothness and hardness criteria, with the aim of keeping an overall balance pleasant to the ear.

The Retain Step The user decides whether or not each new tempo performance should be added to the memory of cases. The newly added tempo performances will be available for the reasoning process in future problems. At this moment *Tempo-Express* has no specific criteria to automatically decide whether to store a newly solved problem.

3 Edit Distance Approach

In general, the edit distance (also known as Levenshtein distance [13]) between two sequences can be defined as the minimum total cost of transforming one sequence (the source sequence) into the other (the target sequence), given a set of allowed edit operations and a cost function that defines the cost of each edit operation. The sequences under comparison are not restricted to consist of quantitative data, and they do not even have to be of the same nature, since the edit operations and their costs can be designed to handle any kind of data.

The set of edit operations used for most purposes contains insertion, deletion, and replacement. Insertion is the operation of adding an element at some point in the target sequence; deletion refers to the removal of an element from the source sequence; replacement is the substitution of an element from the target sequence for an element of the source sequence. Although the effect of a replacement could be established by removing the source sequence element and inserting the target sequence element, the replacement operation expresses the idea that the source and target elements somehow correspond to each other. Ideally, if a source element and a target element are thought to correspond, this is reflected by the fact that the cost of replacing the source element by the target element is lower than the sum of the costs of deleting the source element and inserting the target element. Many other operations can be added, depending on the nature of the sequences, like one-to-many and many-to-one replacements, or transpositions (reversing the order of elements).

The edit distance approach has been applied in a variety of domains, such as text search, molecular biology and genetics. Mongeau and Sankoff [14] have described a way of applying this measure of distance to monophonic melodies (represented as sequences of notes). They extended basic set of edit operations (insertion, deletion and replacement) with *fragmentation* and *consolidation*, i.e. one-to-many and many-to-one replacements respectively. These operations cater for the phenomenon where a sequence of two or more notes (usually with the same pitch) are replaced by one long note of the same pitch, or vice versa.

Phrases that are considered to be melodic variations of each other often contain such kind of variations.

The weights for the edit operations defined by Mongeau and Sankoff take into account the pitch and duration information of the notes. The costs of deletion and insertion are directly related to the duration of the deleted/inserted notes. Replacements are charged by calculating the differences in pitch and duration between the notes to be replaced. The costs of fragmentation and consolidation are similar, where the duration of one note is compared to the summed duration of the set of notes in the other sequence.

3.1 Computing the Distances

Computing the edit distance is done by calculating the minimum cost of transforming a source sequence into a target sequence. This can be done relatively fast, using the following recurrence equation for the distance $d_{m,n}$ between two sequences $\langle a_1, a_2, ..., a_m \rangle$ and $\langle b_1, b_2, ..., b_n \rangle$:

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

for all $0 \leq i \leq m$ and $0 \leq j \leq n$, where m is the length of the source sequence and n is the length of the target sequence. Additionally, the initial conditions for the recurrence equation are:

$$\begin{aligned} d_{i,0} &= d_{i-1,j} + w(a_i, \emptyset) && \text{(deletion)} \\ d_{0,j} &= d_{i,j-1} + w(\emptyset, b_j) && \text{(insertion)} \\ d_{0,0} &= 0 \end{aligned}$$

The weight function w, defines the cost of operations, such that e.g. $w(a_4, \emptyset)$ returns the cost of deleting element a_4 from the source sequence, and $w(a_3, b_5, b_6, b_7)$ returns the cost of fragmenting element a_3 from the source sequence into the subsequence $\langle b_5, b_6, b_7 \rangle$ of the target sequence.

For two sequences a and b, consisting of m and n elements respectively, the values $d_{i,j}$ (with $0 \leq i \leq m$ and $0 \leq j \leq n$) are stored in an $n + 1$ by $m + 1$ matrix. The value in the cell at the lower-right corner, $d_{m,n}$ is taken as the distance between a and b, that is, the minimal cost of transforming the sequence $\langle a_0, ..., a_m \rangle$ into $\langle b_0, ..., b_n \rangle$.

3.2 Optimal Alignments

After the distance value between two sequences has been calculated, it can be easily found out what was the sequence of transformations that yielded this value. To do this, it is necessary to store for each value $d_{i,j}$ ($0 \leq i \leq m$ and

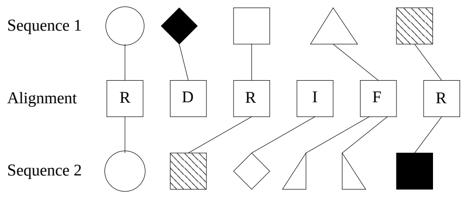

Fig. 2. A possible alignment between two sequences of geometric symbols.

$0 \leq i \leq m$, where m and n are the lengths of the source and target sequence respectively) which was the last operation that was performed to arrive at $d_{i,j}$. From this operation it can be inferred how much each of the indices for the source and the target sequence should be decreased (for example in the case of deletion i is decreased by 1 and j is not decreased). By traversing the matrix in this way from $d_{m,n}$ to $d_{0,0}$, the sequence of operations that yielded the value $d_{m,n}$, can be found.

The sequence of operations that transform the source sequence into the target sequence can be regarded as an alignment between the two sequences, i.e. a correspondence is established between elements in the source sequence and elements in the target sequence, respectively. In the next section we will see that this alignment is the key aspect of the automatic annotation process.

An alignment example is shown in figure 2. For simplicity, the two aligned sequences consist of geometric figures, instead of musical elements. The alignment is shown in between the sequences, as a sequence of operations. The letters in operations stand for replacement (R), deletion (D), insertion (I) and fragmentation (F), respectively. The example shows how matches are made between identical or near identical elements from sequence 1 and sequence 2. When for a particular element in one sequence there is no matchable element at hand in the other sequence, a deletion or insertion operation is performed. A fragmentation occurs between the triangle in sequence 1 and the two half triangles in sequence 2, since the two half triangles taken together, match with the whole triangle. Which of all possible alignments is *optimal* (i.e. has the lowest total cost) obviously depends on the costs assigned to the operations that occur in the alignments.

4 The Annotation Process

To analyze a performance of a melody, a crucial problem is to identify which element in the performance corresponds to each note of the score of the melody. Especially in jazz performances, which is the area on which we will focus, this problem is not trivial, since jazz performers often favor a 'liberal' interpretation of the score. This does not only involve changes in expressive features of the score elements as they are performed, but also omitting or adding notes. Thus, one can normally not assume that the performance contains a corresponding element for every note of the score, neither that every element in the performance corresponds to a note of the score. Taking these performance liberties into account, a description of a musical performance could take the form of a sequence of operations that are applied to the score elements.

From this perspective the edit distance, as described in the previous section, will be very useful. The edit distance has been used before in the performance-to-score mapping problem by Dannenberg [6] and Large [11], among others. The application area has been score-tracking for automatic accompaniment as well as performance timing research. Other approaches to performance-to-score matching have also been proposed. See [7] and [10] for an overview and comparison of several approaches.

The application of the edit distance in the context of comparing performances to scores is somewhat different from the case where scores are comparing to other scores. In the first case, we deal with sequences of different nature. The performance itself is not necessarily a discrete sequence but could be for example a continuous (audio) stream. Although matching score elements to fragments of audio data is not inconceivable, it is probably more convenient to make a transcription of the performance into a sequence of note elements before matching. The resulting sequence of note elements is more appropriate for comparing with scores, but it must be kept in mind that transcription of audio to note sequences is a reductive procedure. For example, pitch and dynamics envelopes are usually reduced to single values.

Another difference between score-performance matching and score-score matching is more conceptual. In score-performance matching, the performance is thought to be *derived* from the score, that is, the elements of the score sequence are *transformed* into performance elements, rather than *replaced* by them. For this reason, in the context of score-performance matching it is more appropriate to talk of *transformation* operations instead of *replace* operations.

4.1 The Edit Operations

The various edit operations can be classified (see figure 3) to make explicit the characteristics of their behavior. Firstly, all the operations refer to one or more elements in the sequences that are aligned. We can distinguish, within this general class of *Reference* operations, those that refer to notes in the score sequence and those that refer to elements in the performance sequence. Deletion operations refer to notes of the score sequence that are not present in the performance sequence (i.e. the notes that are not played), therefore they can be be classified as *Score-Reference* operations. Conversely, insertion operations refer only to elements in the performance sequence (i.e. the notes that were added), so they form a subclass of *Performance-Reference* operations. Transformation, consolidation and fragmentation operations refer to elements from both the score and the performance and thus form a shared subclass of Score-Reference and Performance-Reference operations. We call this class *Correspondence* operations. Figure 3 summarizes the relations between the classes of edit operations.

In our particular case, we are not primarily interested in comparing performance elements to score elements itself, but rather in the changes that are

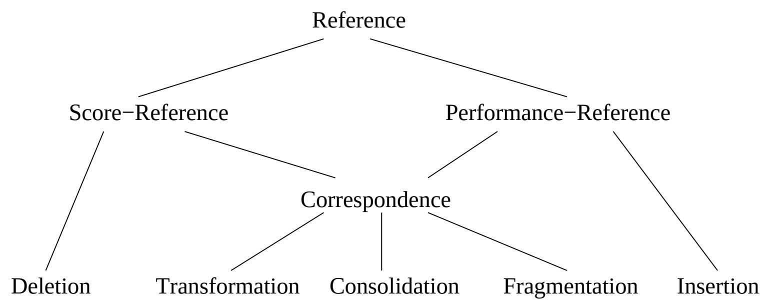

Fig. 3. A hierarchical representation of edit operations for performance annotation.

made to values of score notes when they are transformed into performance elements. Therefore, we view transformation operations as compositions of several transformations, e.g. *pitch transformations*, *duration transformations* and *onset transformations*. Following the same idea, fragmentation and consolidation operations (as described in section 3), can be elaborated by such transformation operations. For example, a consolidation operation could be composed of a duration transformation that specifies how the total duration of the consolidated score notes deviates from the duration of the corresponding performance element.

Based on the fact that the phrases in our data set were played by a professional musician and they were performed quite closely to the score, it may be thought that the performances could be described by only correspondence operations, that map a score element to a performance element, perhaps with minor adjustments of duration and onset. However, as mentioned before, most actual performances, and indeed the performances at our disposal, contain extra notes, or lack some notes from the score. Listening to the performances revealed that these were not unintentional performance errors, since they were often found on the same places in various performances of the same phrase and the effect sounded natural. This implies that in addition to correspondence operations, insertion and deletion operations are also required. Furthermore, we also observed that *consolidation* (as described in the previous section) occurred in some performances. Occasionally, we found cases of *fragmentation*. Other transformations, such as transposition (reversal of the temporal order of notes) were not encountered. Thus, the set of operations shown in figure 3 are basically sufficient for mapping the performances to their scores in our application.

4.2 The Cost Values

Once the set of edit-operations is determined, we need to decide good cost values for each of them. Ideally, the cost values will be such that the resulting optimal alignment corresponds to an intuitive judgment of how the performance aligns to the score (in practice, the subjectivity that is involved in establishing this mapping by ear, tuns out to be largely unproblematic). The main factors that determine which of all the possible alignments between score and performance is optimal, will be on the one hand the features of the note elements that are involved in calculating the cost of applying an operation, and on the other hand the relative costs of the operations with respect to each other.

In establishing which features of the compared note elements are considered in the comparison, we adopted much of the choices made by Mongeau and Sankoff [14]. Primarily, pitch and duration information were taken into to account in order to assign the cost of a transformation operation. For insertions and deletions, only the duration of the inserted/deleted notes is considered. Fragmentation and consolidation both use pitch and duration information. The cost functions w for each operations are given below. P and D are functions such that $P(x)$ returns the pitch (as a MIDI number) of a score or performance element x, and $D(x)$ returns its duration. Equations 1, 2, 3, 4, 5 define the costs of deletion, insertion, transformation, consolidation and fragmentation, respectively.

$$w(s_i, \emptyset) = D(s_i) \tag{1}$$

$$w(\emptyset, p_j) = D(p_j) \tag{2}$$

$$w(s_i, p_j) = |\,P(s_i) - P(p_j)\,| \;+\; |\,D(s_i) - D(p_j)\,| \tag{3}$$

$$w(s_i, ..., s_{i+K}, p_j) = \sum_{k=0}^{K} |\,P(s_{i+k}) - P(p_j)\,| + \; |\,D(p_j) - \sum_{k=0}^{K} D(s_{i+k})\,| \tag{4}$$

$$w(s_i, p_j, ..., p_{j+L}) = \sum_{l=0}^{L} |\,P(s_i) - P(p_{j+l})\,| + \; |\,D(s_i) - \sum_{l=0}^{L} D(p_{j+l})\,| \tag{5}$$

From the equations it can be seen that the cost of transformation will be zero if the score and performance elements have the same pitch and duration, and the fragmentation cost will be zero if a score note is fragmented into a sequence of performance notes whose durations add up to the duration of the score note and whose pitches are all equal to the pitch of the score element.

From a musical perspective, it can be argued that a the cost mapping two elements with different pitches should not depend on the difference of the absolute pitches, but rather on the different roles the pitches play with respect to the underlying harmonies, or their scale degree, as these features have more perceptual relevance than absolute pitch difference. This would certainly be essential in order to make good alignments between scores and performances that very liberally paraphrase the score (e.g. improvisations on a melody) and also in the case where alignment is constructed for assessing the similarity betwee different scores. In our case however, we currently deal with performances that are relatively 'clean' interpretations of the score. As such, changes of pitch are very uncommon in our data. Still, it is desirable to have a more sophisticated pitch comparison approach, to accommodate more liberal performances in the future.

We have also considered incorporating the difference in position in the costs of the correspondence operations (transformation, consolidation and fragmentation). This turned out to improve the alignment in some cases. One such case occurs when one note in a row of notes with the same pitch and duration is omitted in the performance. Without taking into account positions, the optimal alignment will delete an arbitrary note of the sequence, since the deletions of each of these notes are equivalent based on pitch and duration information only. When position *is* taken into account, the remaining notes of the performance will all be mapped to the closest notes in the score, so the deletion operation will be performed on the score note that remains unmapped, which is often the desired result.

The other factor of importance for determining the optimal alignment between score and performance is the relative costs of the operations are with respect to each other. The relative costs of each kind of operation can be controlled simply by adding a constant scaling factor to each of the cost equations above. Experimentation with several settings showed that the score-performance alignments were stable within certain ranges of relative costs. Only when certain values were exceeded the alignments would change. For example, gradually decreasing the cost of deletions and insertions did initially not affect the alignments. Only after reaching a certain threshold, more and more correspondence operations were left out of the alignment and replaced by the deletion and insertion of the involved sequence elements. The stability of the alignments under changing cost functions is probably due to the fact that the performances are generally quite close to the score, i.e. it is mostly unambiguous which notes the performer played at each moment. If the performances had been less faithful representations of the score, it would be less clear whether notes would have just been changed in time and pitch, or rather that some notes have been deleted and others are played instead. This would probably result on a lack of stable alignments. In general, the desired alignments of the different phrases performed at different tempos could be obtained using the same values cost function parameters.

In the calculation of the alignment it is not necessary to take into account low-level operations like onset transformation or duration transformation, because they can be inferred, from the note attributes, once a correspondence between notes in the score and notes in the performance has been established. Also the order in which e.g. a transformation operation is split into pitch transformation, duration transformation and onset transformation, is irrelevant, since all these operations act on the same performance element.

5 Experimentation and Results

Our set of audio data currently consists of alto saxophone recordings of five jazz standards ('Once I Loved' (A.C. Jobim), 'Up Jumped Spring' (F. Hubbard), 'Donna Lee' (C. Parker), 'Like Someone in Love' (Van Heusen/Burke) and 'Body and Soul' (J. Green)), played at various tempos. Each song is played at about 12 different tempos, spread around the original tempo at intervals of approximately 10 beats per minute. These tempos are assumed to cover most of the musically

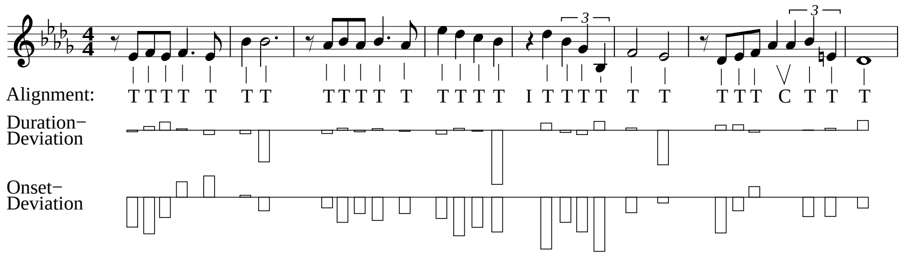

Fig. 4. Graphical representation of the annotations extracted for the first phrase of 'Body and Soul' at 100 beats per minute.

acceptible range of tempos at which the melodies can be performed. The songs were performed by a professional jazz musician, playing along with a metronome. The performer was instructed to stick to the score as much as possible, and play in a consistent expressive way, that is, without intentionally changing the mood. After recording, the performances were segmented into phrases manually, which resulted in a data set consisting of 219 recorded musical phrases. These phrases were transcribed using a melody extraction tool that is being developed by the Music Technology Group (MTG) of the Pompeu Fabra University of Barcelona [8]. The transcriptions are in XML format and comply to the MPEG7 standard for melody description. The note entries in the melody description contain attributes such as the MIDI number of the pitch, onset (in seconds), duration (in seconds) and dynamics. Since the tempo of the performance and the position of the metronome ticks is known, duration and onset values can be easily converted from seconds to metrical beats.

A general analysis of the annotation results, perhaps unsurprisingly, show the extracted annotations showed that more insertions were found in phrases that were performed at slower tempos, and deletions and consolidations more often occurred in faster performances. Closer inspection showed that the inserted notes were mainly ornamental 'leading' notes: notes with a very short duration played 'as an introduction' to the following note. The pitch was often just one semitone above or below the pitch of the note that it embellished. Deletions of notes often occurred in sequences of notes with the same pitch. In some of these cases, the duration of the previous or next note was increased slightly, so it was hard to tell if the note was really deleted, or rather that it had been consolidated together with its predecessor or successor into one large note. Both possibilities could be obtained, respectively by increasing and decreasing the cost of consolidation.

More concretely, table 1 shows the annotations extracted comparing the score and one performance of the first phrase of 'Body and Soul' at tempo 100. The numbers in the table are the deviations with respect to the score. Negative (resp. positive) values for onset transformations mean that the note has been anticipated (resp. delayed). Negative (resp. positive) values for duration transformations mean that the note has been shortened (resp. prolonged). `C5` in the table is a note added by the performer between notes 16-17. `A♭4` consolidates notes 26-27. Empty entries mean that the corresponding transformation was not applied.

Figure 4 shows a summary of the same annotations in a graphical form. The alignment between the score and the performance is represented by the letters T (Transformation), I (Insertion), and C (Consolidation). For the Transformation operations, the duration and onset deviations of performance are shown in bars. The size of the bars show the amount of deviation. Bars above the line indicate a longer duration or later onset, respectively. Bars below the line indicate a shorter duration or earlier onset.

6 Conclusions and Future Work

We have presented a system to annotate cases of musical performances. The annotations are automatically extracted from human performances. This is done by finding the optimal alignment between the score and a transcription of the performance, using the edit distance algorithm.

It should be noticed that even in performances that were intended to be relatively literal interpretations of the score, deletions and insertions of note elements occur. This implies that an annotation scheme that labels performance elements as the one presented in this paper is very useful in music performance processing systems.

This paper has focused on extracting performers' behaviors to annotate cases within the *Tempo-Express* CBR system. The whole system has been only briefly described. Such a system is useful to automatically achieve musically sensible tempo transformations that presently are manually done in audio-processing studios.

At some stages of the annotation process, improvements and elaborations are possible: the hierarchy of edit operations could be specified in further detail, for example by distinguishing between the insertions/deletions of ornamental notes (that often have an intricate relationship to their neighboring notes) versus non-grace notes. Other improvements could be made in the cost calculation of the edit operations. For example, the costs of transformation operations could involve a comparison of the harmonic role of the pitches, rather than the absolute pitch difference. This would be especially valuable in cases where the performance is a liberal paraphrase, or an improvisation on the score.

Acknowledgments

The authors acknowledge the Music Technology Group of the Pompeu Fabra University for providing the XML transcriptions of the performances. This research has been partially supported by the Spanish Ministry of Science and Technology under the project TIC 2000-1094-C02 "Tabasco: Content-based Audio Transformation using CBR".

References

1. Josep Lluís Arcos and Ramon López de Mántaras. An interactive case-based reasoning approach for generating expressive music. *Applied Intelligence*, 14(1):115–129, 2001.
2. Josep Lluís Arcos and Enric Plaza. Inference and reflection in the object-centered representation language Noos. *Journal of Future Generation Computer Systems*, 12:173–188, 1996.
3. Josep Lluís Arcos and Enric Plaza. Noos: An integrated framework for problem solving and learning. In *Knowledge Engineering: Methods and Languages*, 1997.
4. C. Blake, E. Keogh, and C. Merz. *UCI Repository of Machine Learning Algorithms Databases*. University of California. Department of Information and Computer Science, Irvine, CA, 1998.
5. Jaime Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning*, volume 2, pages 371–392. Morgan Kaufmann, 1986.
6. R. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the 1984 International Computer Music Conference*. International Computer Music Association, 1984.
7. P. Desain, H. Honing, and H. Heijink. Robust score-performance matching: Taking advantage of structural information. In *Proceedings of the 1997 International Computer Music Conference*, pages 337–340, San Francisco, 1997. International Computer Music Association.
8. E. Gómez, , A. Klapuri, and B. Meudic. Melody description and extraction in the context of music content processing. *Journal of New Music Research*, 32(1), 2003. (In Press).
9. Maarten Grachten, Josep Lluís Arcos, and Ramon López de Mántaras. A comparison of different approaches to melodic similarity. In *II International Conference on Music and Artificial Intelligence*, 2002.
10. H. Heijink, P. Desain, H. Honing, and L. Windsor. Make me a match: An evaluation of different approaches to score-performance matching. *Computer Music Journal*, 24(1):43–56, 2000.
11. E. W. Large. Dynamic programming for the analysis of serial behaviors. *Behavior Research Methods, Instruments & Computers*, 25(2):238–241, 1993.
12. Fred Lerdahl and Ray Jackendoff. An overview of hierarchical structure in music. In Stephan M. Schwanaver and David A. Levitt, editors, *Machine Models of Music*, pages 289–312. The MIT Press, 1993. Reproduced from Music Perception.
13. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.
14. Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.
15. Eugene Narmour. *The Analysis and cognition of basic melodic structures : the implication-realization model*. University of Chicago Press, 1990.
16. Enric Plaza and Josep Ll. Arcos. Constructive adaptation. In Susan Craw and Alun Preece, editors, *Advances in Case-Based Reasoning*, number 2416 in Lecture Notes in Artificial Intelligence, pages 306–320. Springer-Verlag, 2002.
17. Xavier Serra, Jordi Bonada, Perfecto Herrera, and Ramon Loureiro. Integrating complementary spectral methods in the design of a musical synthesizer. In *Proceedings of the ICMC'97*, pages 152–159. San Francisco: International Computer Music Asociation., 1997.
18. Manuela M. Veloso, Alice M. Mulvehill, and Michael T. Cox. Rationale-supported mixed-initiative case-based planning. In *AAAI/IAAI*, pages 1072–1077, 1997.

Note	Dur. Tr.	Onset Tr.	Pitch Tr.	Ins.	Del.	Cons.	Frag.
1	0.024	0.105					
2	-0.058	0.129					
3	-0.126	0.071					
4	-0.021	-0.054					
5	0.068	-0.075					
6	0.054	-0.007					
7	0.490	0.048					
8	0.051	0.037					
9	-0.031	0.088					
10	0.024	0.058					
11	-0.024	0.082					
12	0.017	0.058					
13	0.061	0.075					
14	-0.030	0.136					
15	0.016	0.106					
16	0.837	0.122					
				C5			
17	-0.110	0.183					
18	0.034	0.088					
19	0.068	0.122					
20	-0.136	0.191					
21	-0.034	0.055					
22	0.535	0.020					
23	-0.078	0.126					
24	-0.085	0.048					
25	0.031	-0.037					
26/27						A♭4	
28	0.000	0.068					
29	-0.030	0.068					
30	-0.150	0.038					

Table 1. Quantified annotations extracted for the first phrase of 'Body and Soul' at tempo 100.