Tree-structured Representation of Melodies for Comparison and Retrieval

David Rizo Valero[1] and José Manuel Iñesta[2]

Departamento de Lenguajes y Sistemas Informáticos, Universidad de Alicante,
Ctra. Alicante-San Vicente S/N,
San Vicente del Raspeig (Alicante) - Spain,
Email: david.rizo@wanadoo.es[1] , inesta@dlsi.ua.es[2]

Abstract. The success of the Internet has filled the net with lots of symbolic representations of music works. Two kinds of problems arise to the user: the search for information from content and the identification of similar works. Both belong to the pattern recognition domain. In contrast to most of the existing approaches, we pose a non-linear representation of a melody, based on trees that express the metric and rhythm of music in a natural way. This representation provide a number of advantages: more musical significance, more compact representation and others. Here we have worked on the comparison of melodies and patterns, leading to motive extraction and its use for the identification of complete melodies.

The success of the Internet has filled the net with lots of symbolic representations of music works. Two kinds of problems arise to the user: the search for information based on content and the identification of similar works. Both belong to the pattern recognition domain.

The applications of variations on the two problems cope from the study and analysis tasks in musicology over musical works to the detection of plagiarism, useful to protect copyrights in the music record industry.

Traditionally music has been represented by means of a set of tupla strings, where each tupla, in diverse ways, usually contains information on pitch, duration and onset time. Both the retrieval and the comparison have been tackled with structural pattern matching techniques in strings, from different points of view, either in the tupla content or in the algorithms [5]

There are some other approaches, seldom applied, like the geometric one, which transforms the melody into a plot obtained tracing a line between the successive notes in the stave. This way, the melody comparison problem is converted into a geometric one [3].

In this paper, we use a nonlinear representation of melody: by means of trees that express the metric and rhythm of music in a natural way. The approach to tree construction is based on the fact that the different music notation figures are designed on a logarithmic scale: a whole note lasts twice a half note, whose length is the double of a quarter note, etc. This representation provides us with a richness of possibilities that the strings never will: more musical meaning and automatic emphasizing of relevant notes, for example. Moreover, the way in

which a string representation is codified strongly conditions the outcome of the string processing algorithms [7], but with trees, this is achieved in a more and intuitive natural way.

We have worked mainly in the comparison of patterns in melodies, which derives in the extraction of musical motives and its later use in the complete comparison for those melodies. The tree distance algorithm of Shasha and Zang [1] has been used.

Although tree comparison algorithms have a higher complexity than the existing methods on strings, the results from the original representation (with no additional or special processing) significantly improve the ones in the same way using strings. This preliminary results open a promising new field for experimentation in a number of applications on the symbolic representation of music.

In this article we will develop the tree construction, as much as a set of rules necessary to deal with the complexity above described and its later use in melody comparison by content.

Firstly, the method for tree construction will be presented and how it deals with the notation problems that may appear. Secondly, a procedure for tree simplification is described. Then, the method for comparison and the results are presented, and finally conclusions are stated.

1 Tree Construction Method

As described above, the tree construction is based on the logarithmic relation among duration of the different figures. A sub-tree is assigned to each measure, so the root of this sub-tree represents the length in time of all the measure. If just a whole note is found in the measure, the tree will consist of just the root, but if there were two half notes, this node would split into two children nodes. Thus, recursively, each node of the tree will split into two until representing the notes actually found in a measure (see Fig. 1).

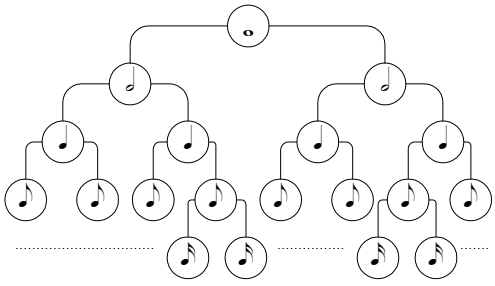

Fig. 1. Duration hierarchy

The representation of a melody will be made in the following way. Each leaf node represents a note or a silence, keeping the pitch as the label of the node.

Silences will be kept by means of an special label that identifies the node as a silence. Each node has an implicit duration according to the level of the tree in which it appears: a whole note in the root, half notes in the second level, quarter notes in the third, and so on. It will be necessary to put intermediate nodes until reaching the length of the note to be represented. Only the leaf nodes will contain a label value.

In addition to the duration of the notes, by means of the left to right ordering of child nodes we are also establishing the time in the measure in which they begin to play. In the third tree level, that corresponding to quarter notes, we can find up to four subtrees, one for each beat in a measure in the 4/4 meter. The leftmost subtree holds the first beat of the measure, the second subtree is representing the second beat of the measure, the third is equal to the third beat and the rightmost one keeps the remaining beat. As each note is held on a leaf node it will be necessary to develop the branches of the tree by means of binary subdivisions until getting a node that lasts the same as the note figure we want to express.

An example of this scheme is presented in Fig. 2. The left child of the root has been split in two subtrees to represent the quarter note C. This one will occupy the time space that represents the node leaf in which it is, exactly a beat in the measure. In order to represent the durations of the eighth notes it will be necessary to unfold a level more. The half note F onsets at the third beat and, as it occupies two beats, its position is in the second level of the tree. Note that the figures that appear in the inner nodes are there only for help purpose, in the real trees we only have the pitch as a label for leaf nodes.

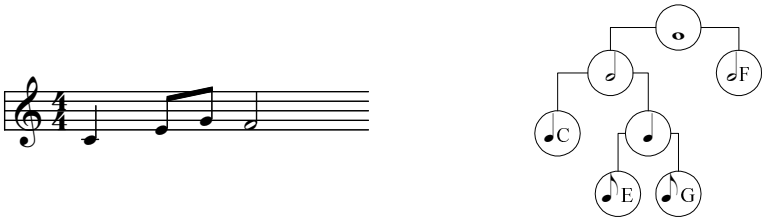

Fig. 2. Simple example of tree construction.

In some occasions the situation can be more complicated. For example, what happens when the duration of a note is greater than that of the half corresponding subdivision? Musical notation uses small points beside the notes for that purpose. In this situation, a note can not be represented only by the complete subtree in which it onsets. It is well known that the ear does not perceive in a very different way a whole C note from two half C notes played one after the other, even more if the interpreter play them *legato* [11]. Thus, when a note exceeds the proper duration, we will subdivide it in order to complete the time of the note by means of nodes in sub-trees enough to complete the duration of the note with smaller pieces. See Fig. 3 for an example. There, the C has been

subdivided into two notes, firstly occupying a time of quarter note, and then an additional eighth note.

Fig. 3. Notes exceeding its proper length.

The tied notes will been represented in the same way (see Fig. 4), breaking the tie in the tree representation.

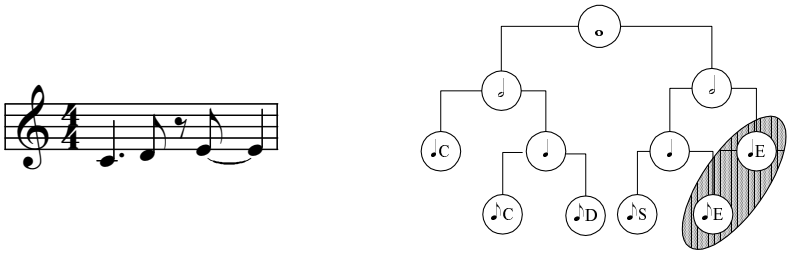

Fig. 4. Tied notes. S stands for "silence".

If it were necessary, the dotted and tied notes could be marked to represent this special situation.

1.1 Processing of Other Meters and Non Binary Structures

Next we will see how to deal with melodies written in non quaternary meters, as well as non binary rhythmical structures that include ternary meters and compound ones.

2/4 and 3/4 Meters When we face up with 2/4 meters the proper approach is to use the same technique as for 4/4 meters, with the difference in the duration of the root node, that correspond in this case to a half note.

The case of ternary meters is very similar. Once again, the only difference from the 4/4 case is the duration of the root and the fact that the subdivision of the root is not binary. This is because the natural way of splitting a 3/4 measure is not into two dotted quarter notes, but three quarter notes. See Fig. 5.

Fig. 5. Ternary meters.

Different Beat Durations The problem would arise when comparing two scores representing the same music but written with different beat duration (for example, 4/4, 2/2 or 8/8, see Figs. 6 and 7 as an example). It can be observed that the structure for both trees is the same, regardless of the different music notations. This example illustrates the fact that using this approach one does not need to think in terms of musical notation but just in terms of tree levels that inherently have a time and rhythm meaning.

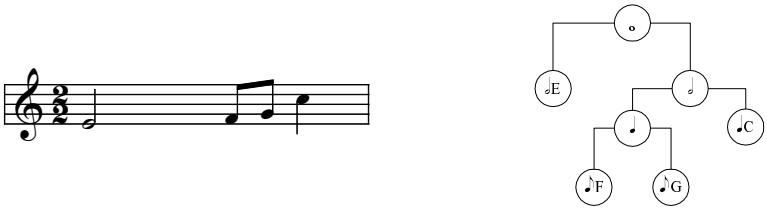

Fig. 6. Difference in beat duration, see equivalence in Fig. 7.

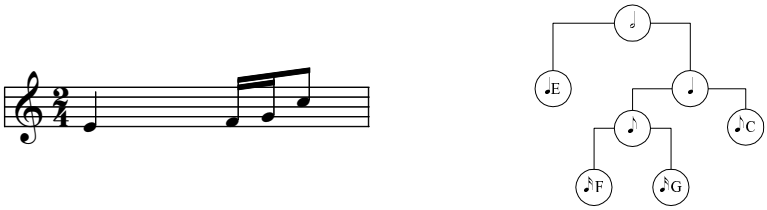

Fig. 7. Difference in beat duration, see equivalence in Fig. 6.

Compound Meters In compound meters the binary tree representation can not be applied in a natural way since the level that makes explicit the beat division of a measure must be ternary. The key point here is to note that the root contains the length of the measure, and there is one child for each compound time. In the Figs. 8 and 9 two representative cases are presented that can be a prototype for all the others.

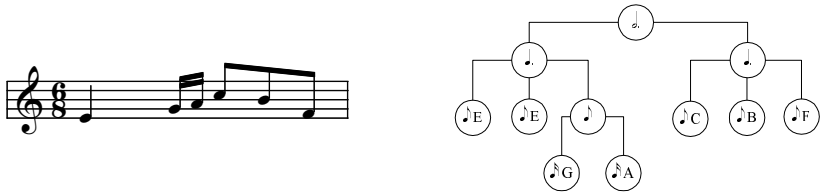

Fig. 8. Compound meter.

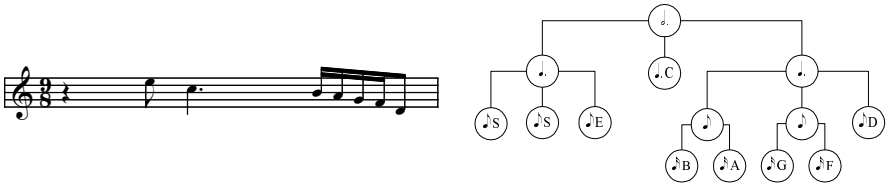

Fig. 9. Compound meter.

Tuplets In groupings like triplets the technique utilized is not the same as in compound meters. The interpretation of a quintuplet of eighth notes can be translated approximately as a sequence of three dotted 16th notes (see Fig. 10 for illustration). Applying this, if this situation is found in a binary measure, it will be represented like it was the second situation, that can be represented binary, rather than creating five children for that node. This situation can be applied to all sorts of tuplets.

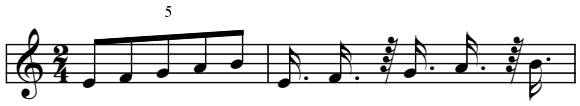

Fig. 10. Tuplet transformation. Second meter is the transformation of the first one

1.2 Adornment Notes

The first thing we could do is to ignore them, because their function is just to adorn, as their name indicates. See transformation of Fig. 11 in 12. But, as in most situations we may have not explicitly indicated that a note is an adornment note, it is easier and better to represent them in the same manner as tuplets, just indicating its interpreted (played) value. See the grace note at the measure in Fig. 11. It can be carried out in three ways according to whom the length is reduced to make the adornment: the note coming next, the previous one, or to both. The following three examples (Figs. 13, 14 and 15) illustrate these

situations. The duration is taken from the note of adornment as a 32th note, that, as being much smaller than the note to which it accompanies, is enough for our purpose.

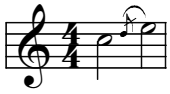

Fig. 11. Adornment notes: grace note.

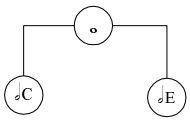

Fig. 12. Removing of the grace note of 11.

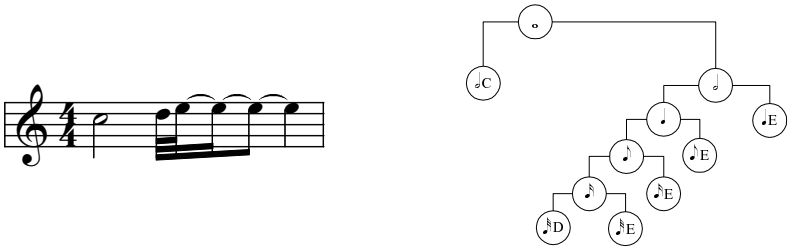

Fig. 13. The grace note reduces the previous note.

Dealing this way with so the adornment notes leads us to be able to read a MIDI file, where no notation showing this kind of notes exists, without problems. In the next sections we will explain how to deal with this and other artifacts for making the aimed comparison.

Trills We choose the trill to illustrate the rest of adornments that can affect a note. Its transformation into tree will require a previous conversion of the notes that are really interpreted, as it was done with grace notes. See Fig. 16.

1.3 Representation of Complete Melodies

The presented method places each measure in a different tree. The representation of a complete melody requires to establish how to group the measures. Initially

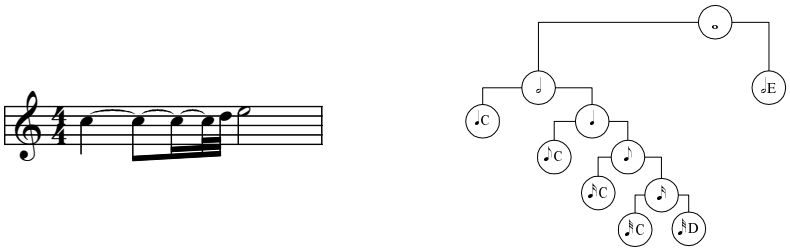

Fig. 14. The grace note reduces the next note.

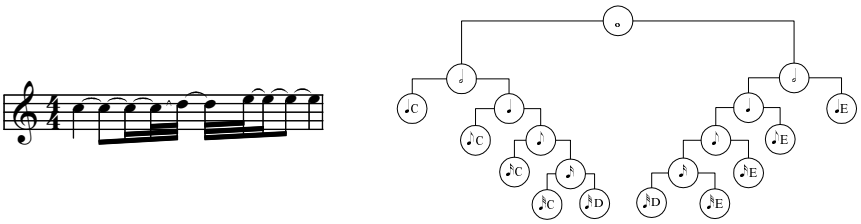

Fig. 15. The grace note reduces both the previous and the next note.

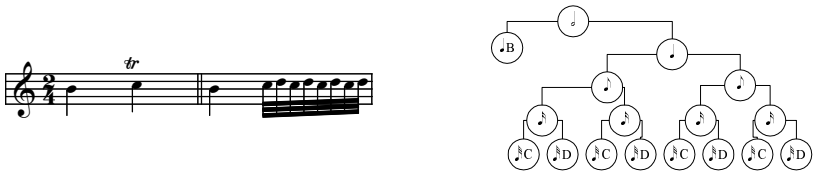

Fig. 16. The trill in first measure is transformed into the notation shown in the second measure.

we could group them two by two, repeating this operation until obtaining a single tree (see Fig. 17).

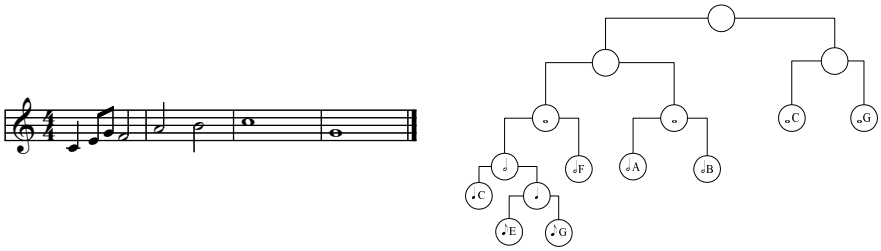

Fig. 17. Complete melody hierarchically constructed.

Nevertheless, trees would grow in height very quickly this way. The number of nodes of the tree would be multiplied and it would make the tree comparison algorithms very time consuming. Another approach it could be to establish a hierarchy criteria that results in only one tree, like the GTTM hierarchy [6] without any node reduction, but this is not an improvement for that purpose.

In our method, we have chosen to build a tree with a root for the whole melody and each measure is a child of the root (Fig. 18). Thus, the level of the tree only grows in one when representing melodies from the case of single measures. This is like having a forest but linked to a common root node in a single tree.

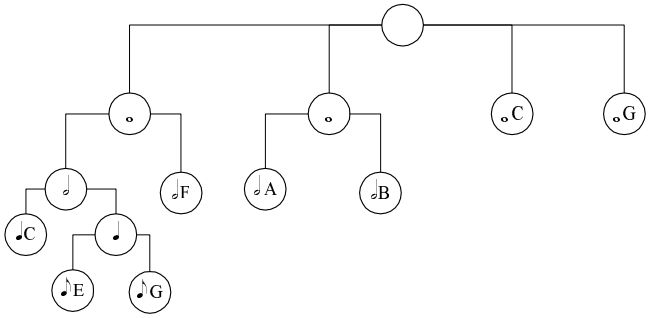

Fig. 18. The way to represent all measures in a single tree.

2 Bottom-up Propagation of Labels

The main objective of this work is to develop a system to identify similarities between two melodies represented as trees. So we need to compare melodies,

not to represent a given interpretation of one melody (as it could be filed in an electronic format) with the highest detail. Therefore, any technique that simplifies or increases the efficiency of the comparison will be valid.

It is not difficult to realize the complexity achieved by the trees when the rhythmical structure does not agree exactly with the successive subdivisions of the binary tree, for example when reading a MIDI file, where notes were sequenced played by a human interpreter and slight variations on the score occurs. This implies a greater temporary and space cost in the algorithms as well as a difficult identification of equivalent notes in two different interpretations of the same score. Our goal is to be able to represent melodies in a reduced format able to keep the main features of the melody. For this, the trees need to be pruned and some notes will propagate upwards according to musicology criteria.

The fact to promote a pitch label upwards implies that the note in that node is more important than that of the sibling node, because any selection criterion have to drop the less important notes. The propagation criteria proposed here are based on the fact that, in a melody, there are notes that contribute more than others to its identity.

The rules for the reduction of the tree are (remind that the node labels are tone pitches):

R1 Given a node N with two children X and Y, if one of those children (X) contains the same label that the brother of node N, the other son (Y) is promoted. Thus more melodic richness is represented instead of extending more the notes we have. When dealing with nodes with more children the case is the same, but it only works if there are several X with the same label.

R2 In case that all children of a node have the same label, they are deleted and its label is placed in the upper node. Thus, two half C notes are equivalent to a whole C note. Mongeau and Sankoff call this "expansion propose and compression"[4].

R3 If one of the brothers lasts at least one eighth part of the original duration (before being propagated) in relation to the other brothers (it has arrived here propagated from three levels down) the brother of greater original duration is chosen. Thus we avoid very short notes (adornment notes) having more importance than long notes [1].

R4 When various nodes are equivalent in original duration or when promoting a note implies losing the other we will choose not to propagate anything, because by promoting nodes we do not contribute with anything new.

R5 Silences never have greater precedence than the notes.

R6 In case that there is only a child (either because of the tree construction or by propagation) it is automatically upgraded.

2.1 Swing Time Treatment

Next we will see the equivalence between a melody interpreted as the score (Fig. 19) and its version with swing (Fig. 20).

[1] this difference of one eighth of the duration has been established in a totally empirical way

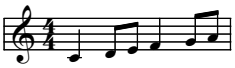

Fig. 19. Melody without swing.

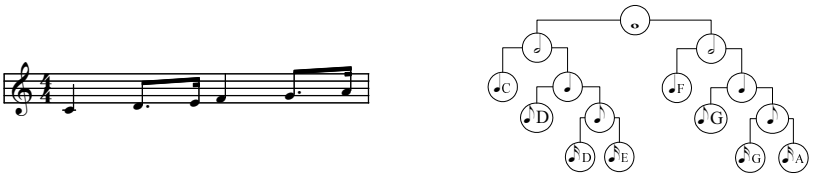

Fig. 20. Same melody as in Fig. 19 interpreted with swing.

In Fig. 21 it can be observed how the transformation by means of the rules exposed above leads the tree to have the same structure as it would have if it would be built from the original score in Fig. 19. The labels E and A ascend by the rule R1. More propagations are not made (rule R4) because if we apply them some notes would be lost (the key E).

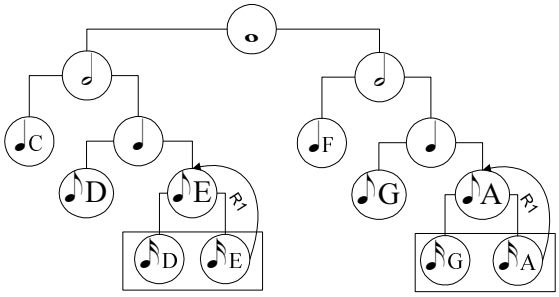

Fig. 21. Propagation of the keys of the tree with swing. Nodes into the rectangles will disappear after the propagation.

2.2 Adornment Notes

Figure 22 shows the application of the propagation rules on the score of Fig. 13. With the other possible interpretations of the adornment note (see figs. 14 and 15) the result is the same. The important part of melody is retained, deleting the adornments.

3 Editing Distance between Trees

We can define the editing distance between two trees like the minimum cost of the sequence of operations that transforms a tree into the other [1].

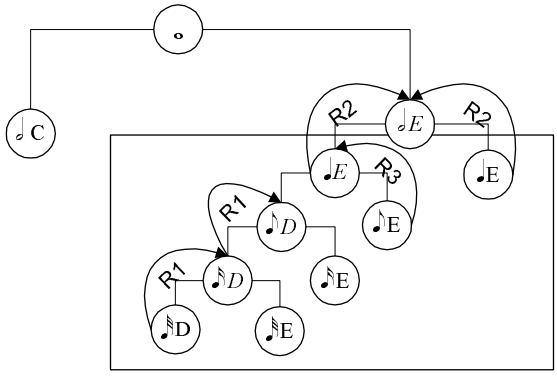

Fig. 22. Propagation of keys in the interpretation of an adornment note (figure 13).

The editing operations are equivalent to those used in the string edition: deletion of a node, insertion and substitution of the label of a node. Each kind of operation has an associated cost. The difference is based in how it is left the tree after applying it an operation.

Substitution The key of a node is changed by the one of the other. The cost of this operation will brought by a function on the keys of both nodes.

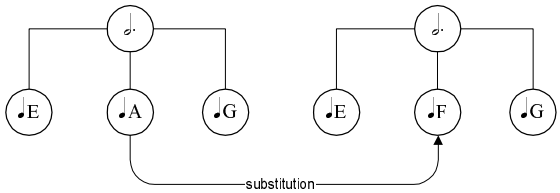

Fig. 23. Change of label A by label F.

Insertion A new node is added to the tree in a given point. The cost is related to the operation itself and not to the label that it contains. Certain children of the node in which the new node is inserted will be changed to be children of the new node.

Deletion A node of the tree is eliminated. Like in the insertion, the cost comes implicit in the operation and it does not depend on the key of the node. The children of the node are hanged to be children of the parent of the erased node.

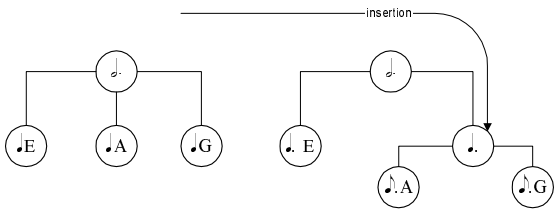

Fig. 24. Insertion of an inner label

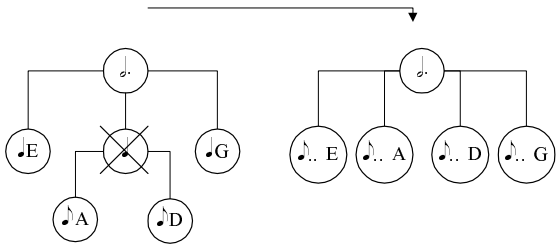

Fig. 25. Deletion of internal key.

Musical Sense of Editing Operations As we have seen, the deletion and insertion of nodes in a tree are not trivial matters, and then its necessary to know the musical meaning they imply.

In case of substitution, there are two things to be compared, the pitch kept as the value of the node, and the depth of the node, that indicates the duration, the figure of the note. In our experiments we have only took care of the pitch, and not of the node depth. This is because in the tree editing distances we do not only have to align some nodes, but all of them. This way, to have a match of two nodes in the two trees the labels must both be the same and must have the same number of ancestors in both cases. This guarantees that the structure and the form of the tree are taken into account and therefore the rhythm.

The operation of insertion is somewhat different. As seen, the insertion of a node implies making it a parent of others. As one node cannot have no siblings, we take this operation as an intermediate step in the comparison algorithm.

In the deletion of nodes, the children of the deleted node are converted into siblings of the siblings of the deleted node. It is the same as if we change the rhythm a beat or less in a measure.

What is important here is to note that to arrive to an alignment, a null difference between trees, the more similar the rhythmical structure (tree structure) the trees have, the less operations of deletion and insertion have to be done, and the smaller difference between them is achieved.

4 Motive Extraction

The motive identification is to music the same as pattern finding in the computer science. There have been approaches to this problem from the string pattern matching like [8].

Applying tree pattern discovery methods [1] to our melody representation on trees should give us the extraction of important patterns or motives.

As the results of pattern discovery are subtrees representing motives, to get the distance between two different melodies, we could compare forests of subtrees that act as features of each melody, avoiding the comparison of full trees, and therefore, increasing the performance of the algorithm.

5 Experiments

In order to test the effectiveness of our method we have compared it with methods of string comparison. We will deal with melodies as a string of symbols codifying pitches, pitches and length, contours or intervals. Since the tonality problems [4] can be solved in the same way for strings and for trees, in order to make a more objective comparison we will compare initially melodies in the same key. It is important to keep in mind that our method introduces in an implicit manner the representation of the rhythm. Because of this, when two melodies differ only in rhythm the best performance will be achieved for the codification independent of this feature. On the other hand, when two melodies are exactly equal in rhythm but changes the pitches of notes, if the duration is not observed, this similarity will not be perceived.

The experiments have been make with 22 melodies from MIDI files downloaded from the Internet with no manual edition. Only the melody track has been compared. In order to make comparisons regardless of musical keys, we have considered pitch intervals as labels instead of absolute pitches. We will only show the significant results. The editing distance functions have been the same for all cases:

Insertion 1.0
Deletion 1.0
Substitution 0.0 if the interval/note is the same, 1.0 otherwise, and only for the tree algorithms, *infinity* if the comparison is between an inner node label and a real note (a leaf node)

The comparisons have been made to the complete melodies without using any windowing mechanism.

For each melody we also have recorded at least two or three variations in the performance. We have compared each pair of melodies and then have ordered the distances weighted by notes number. For each melody, the recognition is successfull if the melody having the minimum edition distance is one of the variations of that melody.

In the graphic in Fig. 26 we show the recognition percentages using different melodies representations:

ST String of tuples of interval and duration.
NR Trees constructed as the proposed method with no propagation between nodes
FR Trees fully pruned until no possible reduction rule applies.
L5 Trees pruned until level 5, that means a precision of 16th notes.
L6 Trees pruned until level 6, that means a precision of 32th notes.
L7 Trees pruned until level 7, that means a precision of 64th notes.

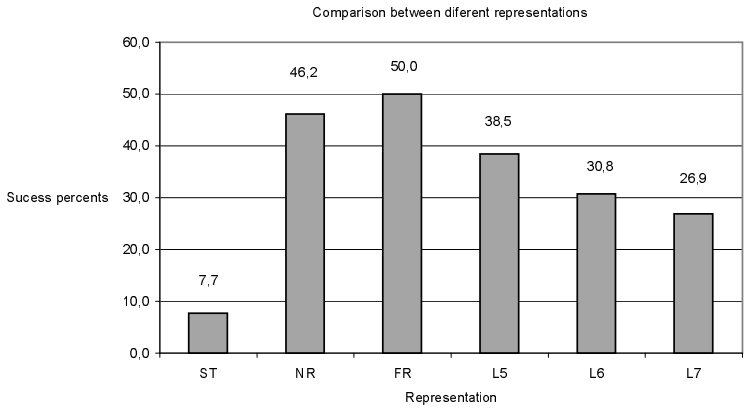

Fig. 26. Comparison with different representations

The errors are mainly caused by the fact that the new algorithms has focused more on rhythmical issues, rather than on pitch differences. The improvements done on pitch distances on strings can be directly applied to trees. Note the simplicity of the substitution function, and therefore, the richness it can bring to the melodic distances.

It is very important to observe that we have experimented directly with different interpretations of the same melody. The examples are real performances. Usually, this kind of experiments work much better with quantized melodies or MIDI files generated directly from scores [10].

6 Conclusions and Future Work

In this work our aim was to show the richness of the tree representation for melody and the applications that can be derived. The results obtained with no additional processing for strings or trees show these possibilities.

We have tried to get the minimum editing distance, based only on pitch. An algorithm has been introduced to prune the original trees through a bottom-up propagation of node labels according to musicological criteria. We have verified that propagated and pruned trees work better than the original ones.

Our method has focused only in rhythmical matters, not in analytical hierarchies. As a future work, the same tree editing algorithms could be applied to trees result of the GTTM [6] analysis.

The addition of polyphonic scores [9] is not a difficult matter in our representation. The only thing must do is let the node labels to contain various pitches. In this situation, the work has to focus on the substitution function definition in the editing distance.

Acknowledgments

This work has been funded by the Spanish CICYT project TAR; code TIC2000–1703–CO3–02

References

1. Shasha S., Zhang K. Approximate Tree Pattern Matching. Pattern Matching Algorithms (1997) 341–371 Oxford University Press
2. Apostolico A. On-line string searching Pattern Matching Algorithms (1997) 90–122 Oxford University Press
3. Doncha Ó Maidin. A geometrical algorithm melodic difference Computing in Musicology (1998) 65–72 MIT Press
4. Mongeau M., Sankoff D. Comparison of musical sequences Computers and the Humanities **24** (1990) 161–175
5. Smith L.A., McNab R.J., Witten I.H. Sequence-Based Melodic Comparison: A Dynamic-Programming Approach. Melodic Similarity. Concepts, Procedures, and Applications (1998) 1001–117 MIT Press and Center for Computing in the Humanities (CCARH), Stanford University
6. Lerdahl F. , Jackendoff R. A Generative Theory of Tonal Music (1983) MITP Cambridge, Massachusetts
7. Cruz-Alcázar P.P., Vidal-Ruiz E., Learning Regular Grammars to Model Musical Style: Comparing Different Coding Schemes Proceedings of the 4th International Colloquium on Grammatical Inference (ICGI-98) **1433** 211-222
8. Cambouropoulos E., Crochemore M., Costas S., Iliopoulos, Mouchard L., Pinzon Y.J. Algorithms for Computing Approximate Repetitions in Musical Sequences (1999) Austrian Research Institute for Artificial Intelligence
9. Lemström K., Tarhio J. Searching Monophonic Patterns within Polyphonic Sources Proc. 1998 International Computer Music Conference (ICMC98) **2** 1261–1279
10. Mitzenmacher M., Owen S. Estimating Resemblance of MIDI Documents ALENEX (2001) 79–90
11. Cope D. Experiments in Music Intelligence Proc. Int. Computer Music Conf. (1987) Computer Music Association
12. Chih-Chin Liu, Jia-Lien Hsu, Arbee L.P. Chen", Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases IEEE International Conference on Data Engineering (1999) 257–286