Incremental Parsing for Real-Time Accompaniment Systems

Giordano Cabral[1], Jean-Pierre Briot[1], François Pachet[2]

[1]Laboratoire d'Informatique de Paris 6 – Université Pierre et Marie Curie
8 Rue du Capitaine Scott 75018 Paris – France

[2]Sony Computer Science Lab Paris
6 Rue Amyot 75005 Paris – France

{Giordano.CABRAL,Jean-Pierre.BRIOT}@lip6.fr, pachet@csl.sony.fr

Abstract

The incremental parsing (IP) algorithm has been successfully used in real-time applications, due to its efficiency in modeling musical style. For example, musical systems using IP are able to continue the musical phrases played by a musician in a consistent way. This paper proposes some modifications to the original IP algorithm in order to allow its use in accompaniment systems.

1. Introduction

Statistical analysis techniques have been applied to musical material with the aim of modeling musical style. Dubnov (Dubnov et al. 1998) demonstrated how Markov chains can be used to learn the dynamic behavior of melodies from a database of examples. Lartillot (Lartillot et al. 2001) used these probabilistic models to classify musical styles, while Pachet (Pachet 2002) used them to create an interactive system: The *Continuator*. We are interested in using similar approaches in a real-time accompaniment system.

Our work is strongly inspired by the *Continuator*, and intends to extend its methods to with the domain of accompaniment. The *Continuator* is able to continue the phrases played by a musician, coherently to their style, by computing in real-time a variable-order Markov model of the corpus. In order to efficiently compute the model, the system uses an algorithm called Incremental Parsing (IP) (Ziv and Lempel 1977), which was initially designed for the LZ compression method. The *Continuator* explores several interactive modes, from the simple question-answer (in which the system merely responds to the musician's input) to the collaborative one (in which the system plays continuously, restraining the generated sequence to harmonically acceptable notes). Despite the unquestionable capacity of the system to emulate the style of the musical input (see some video excerpts in http://www.csl.sony.fr/~pachet/continuator) in the question-answer mode, the system demonstrated to be deficient in performing accompaniments. In fact, the difficulty relies in balancing the ability of generating stylistic coherent musical phrases and adapting it to the musician's input. This work proposes some modifications to the IP algorithm hoping to find a better way of dealing with the problem.

Next section exposes the core problem involved in developing our accompaniment system, identifying possible tradeoffs among adaptation, continuity, and prediction. Section 3 describes the traditional Incremental Parsing used, for instance, in the *Continuator*. Section 4 presents the consequent modifications in the original IP. Finally, Section 5 draws some conclusions and presents our future works.

2. Musical Accompaniment

The system we are developing must be capable of accompanying a user who sings a melody via the computer microphone, without any previous knowledge of the song. The accompaniment style and the tempo are previously defined. For the first prototype, Brazilian Bossa-Nova style, at 90 bpm. These constraints simplify the problem: the system must only be able to find the best chords for a melody that is currently being sung. In this context, we can consider the application as performing real-time harmonization.

The core problem involved in the development of such a system is to establish a tradeoff between continuity and adaptation, as we can see in Figure 1. On the one hand, the chords chosen for the accompaniment must keep certain continuity. It means that the choice of the chords to be used depends on the previous ones. On the other hand, the chosen chords must fit the melody being sung.

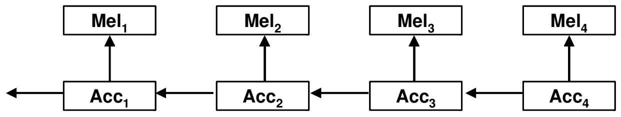

Figure 1 – dependency relation between accompaniment and leading melody. Chords depend both on previous chords and on concurrent melody.

Moreover, as we are in a real-time scenario, there is one extra requirement: the capability of predicting notes, since the melody that will be sung is unknown. In fact, we observed 3 strategies employed by accompaniment musicians, related to different combinations of adaptation, continuity, and prediction.

1. *Prediction/Retrieval* – given a sequence of notes (melody) sung by the singer, the musician imagines (predicts) a continuation, and tries to find the good chords to play along with. The adaptation relies on its ability to combine both melody and harmony, and the continuity is maintained while the singer follows a melody sufficiently similar to the predicted one. At each moment, the musician reviews the coupling harmony/melody, updating or rebuilding the predictions since conflicts are found, or the predicted phrase ends.
2. *Retrieval/Continuation* – according to the sung melody, the musician creates an appropriate accompaniment and subsequently tries to find a continuation for it. The continuity is naturally respected, but the musician might constantly review the coupling harmony/melody, and restart the process in case of conflict.
3. Joint – melody and harmony are seen as a whole. Since chords are groups of notes, it is reasonable to consider the ensemble note and chord also as a chord, thus adaptation, continuity, and prediction are simultaneously taken into account.

The IP algorithm can be extended in order to operate according to these 3 behaviors. Next sections explain its original and the modified versions.

3. Incremental Parsing

This algorithm is originated from the analysis phase of the Lempel-Ziv (Ziv and Lempel 1977) compression method. The technique can be applied to music, whether interpreted as a sequence of notes. The parsing is divided into 2 phases. First, the input sequence is read, generating a model that captures the redundancy. Second, a compressed representation of this sequence is encoded. For interactive systems, the model captures probabilities of transitions between notes, and a stochastic simulation of that model, for a given new sequence, substitutes the second part. Normally, inverted suffix trees, (i.e. which sequences are read from the leaves to the root) are used, and the indexes of the continuations are stored in each node.

Figures 2 and 3 illustrate how the algorithm works for interactive systems. The IP incrementally reads the input sequences, where each sequence is divided into an occurrence and a continuation. Supposing the first sequence is [C, A, F, E, the possible occurrences and continuations might be [C, A, F] and [E], [C, A] and [F], or [C] and [A] (considering we are interested in one-sized continuations). At each cycle, the shortest pattern that does not exist yet in the dictionary of sub patterns is chosen and added to the model, as well as its continuation. The subsequence [C, A, F] is read, having 4 as the continuation index, which implies that each node has the continuation index 4. Subsequently, the [C, A] subsequence is read, resulting on a continuation index 3. Finally, [C] subsequence is read, creating the last branch of the suffix tree (the complete tree at this stage is presented in Figure 2).

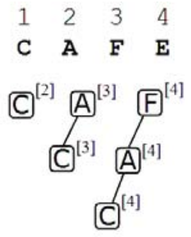

Figure 2 – suffix tree generated by the original IP for the sequence [C, A, F, E].

Later on, a second sequence ([C, C, F, G]) is observed. The same process is applied. For example, the branch [C, C, F] is attached to the branch [C, A, F], since they have the same suffix ([F]). The final tree is shown in Figure 3. This mechanism allows only new information to be added, avoiding the storing of redundant data. In order to search a continuation for a new sequence, one just needs to browse the tree, looking for the longest suffix. For instance, for the sequence [E, A, F], the answer is [E] (continuation index 4), related to the subsequence [A, F] (as demonstrated in gray in Figure 3).

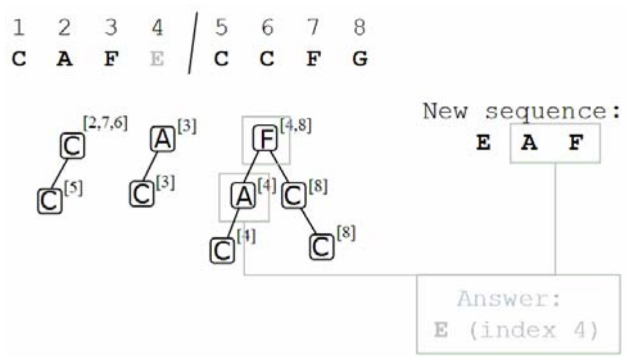

Figure 3 – suffix tree generated by IP for the sequences [C, A, F, E] and [C, C, F, G], and possible query. In gray, the answer for the query.

4. Modified Incremental Parsing

This section presents the necessary modifications for allowing the IP to present the behaviors cited in Section 2. In the first case (Prediction/Retrieval), we propose to modify the structure of the suffix tree (Figure 4). Instead of storing possible continuations, it would store the possible accompaniment chords. In the second case (Retrieval/Continuation), we propose to create 2 distinct suffix trees, one for the lead melody, and the other for the accompaniment (Figure 5). Each leaf in the melody tree would point to a node in the accompaniment tree. In the third case (Joint), we propose to mix the states. The elements in the sequence would neither be a single note (from the leading melody) nor a chord (from the accompaniment), but rather a tuple <note, chord>, as shown in Figure 6.

We can examine how that would work by making use of an exampleSupposing that the sequences used in the previous example ([C, A, F, E] and [C, C, F, G]) are accompanied respectively by the chords [Am, %, F, E7] and [Am, F, E7, %], where "%" indicates that the chord remains the same. After a while, a new sequence [E, A, F] is observed.

Case 1 is illustrated in Figure 4. The tree is very similar to the original IP tree, but the indexes refer to the chords of the accompaniment, and not to possible continuations. The longest suffix of [E, A, F] found in the tree is [A, F], resulting on the index 4. The retrieved chord is, thus [E7].

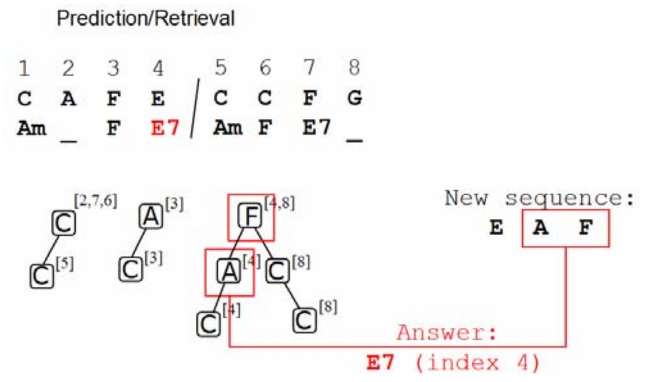

Figure 4 – prediction/retrieval behavior.

Figure 5 illustrates the behavior of Case 2. On the top-left corner, there is the suffix tree related to the accompaniment. On the bottom-right corner, there is the suffix tree related to the leading melody. As in the previous case, the longest suffix of the new sequence [E, A, F] is [A, F]. However, this suffix returns a subsequence of chords (in the example, the chords between indexes 2 and 3, i.e. [Am, F]). The accompaniment suffix tree is then used in order to search a continuation. In this case, there would be 2 possibilities (to be randomly chosen): 4 or 7.

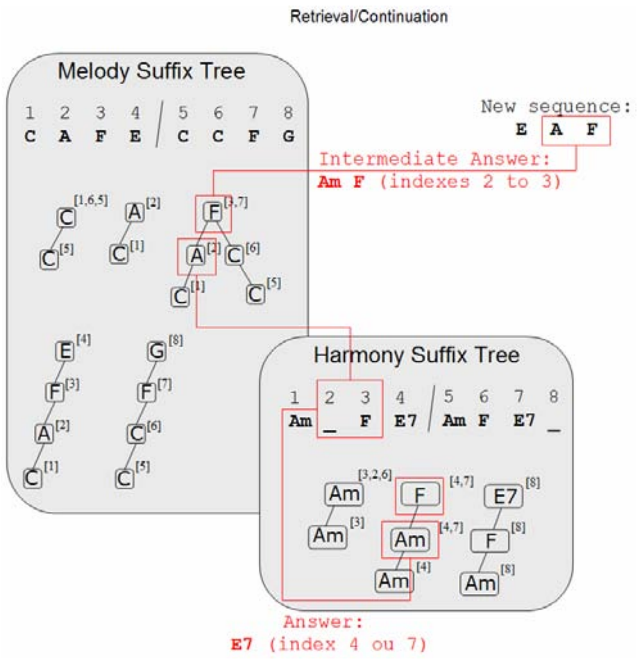

Figure 5 – retrieval/continuation behavior.

In case 3, each node is a tuple <note, chord>, thus the chords are also considered in the query. Let's assume the system has played the chords [%, C7, F] along with the melody [E, A, F]. The query becomes [<E, %>, <A, C7>, <F, F>] (Figure 6). In such a case, the longest suffix found is not of size 2 anymore (as the previous [A, F]), but of size 1 (<F, F>), demonstrating this strategy is less likely to find solutions.

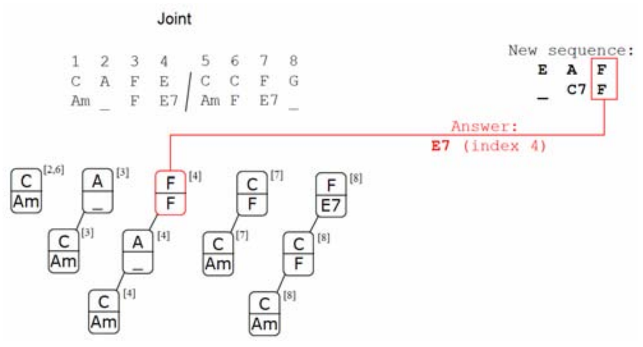

Figure 6 – behavior of joint solution.

5. Discussion and Future Work

We consider that the 3 suggested IP variations reflect the 3 behaviors mentioned in Section 2. In the first one, the system predicts a continuation to the leading voice melody, and then searches an appropriate accompaniment. When the actually sung melody conflicts with the predicted one, the system is restarted with a new query. In the second one, the system searches an appropriate accompaniment for the current leading melody, and then generates a continuation for the sequence of chords. When the sung melody conflicts with the chosen accompaniment, the system is restarted with a new query. In the last one, the system considers note and chord as a whole, and searches an adequate continuation for both simultaneously. A situation where the query does not get any answer would be considered as a conflict between leading melody and accompaniment.

Such algorithms are currently being implemented, and we intend to run some experiments with musicians, in order to evaluate the strengths and weaknesses of each one, and to investigate whether the system really emulates musician's abilities. Obviously, it will be difficult to measure the quality of the system, given its subjectivity. However, we believe that by computing the time of use of each mode, recording and analyzing users reactions, and asking for their feedback, it will be possible to conceive a hybrid optimized model, combining the strengths of all three algorithms.

6. Conclusion

In this work, we proposed modifications in the Incremental Parsing algorithm in order to make it work with concurrent sequences. Thus, it would be suitable to musical accompaniment systems. We suggested 3 variations, emulating 3 different behaviors of musicians: prediction/retrieval, retrieval/continuation, and joint. We are currently analyzing possible criteria to evaluate these algorithms, in order to provide comparative data.

Acknowledgements

We would like to thank the Sony Computer Science Lab in Paris for their support.

This research is supported by CAPES/COFECUB, Brazil/France.

References

Lartillot, O., Dubnov, S., Assayag, G., and Bejerano, G. 2001. Automatic Modeling of Musical Style. In *International Computer Music Conference*, La Havana.

Dubnov, S., Assayag, G., and El-Yaniv, R. 1998. "Universal Classification Applied to Musical Sequences". In *Proceedings of International Computer Music Conference*, pp. 332-340.

Pachet. P. 2002. "The Continuator: Musical Interaction with Style". In *ICMA, editor, Proceedings of ICMC*, pp. 211-218.

Ziv, J., and Lempel, A. 1977. "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.