DYNAMIC CHORD ANALYSIS FOR SYMBOLIC MUSIC

Thomas Rocher, Matthias Robine, Pierre Hanna, Robert Strandh

LaBRI
University of Bordeaux 1
351 avenue de la Libration F 33405 Talence, France
simbals@labri.fr

ABSTRACT

In this paper, we present a new method for chord recognition from symbolic music. This method builds a graph of all possible chords and selects the best path in this graph. A rule-based approach is adopted to enumerate chord candidates from groups of notes by considering compatible chords and compatible keys. The distance proposed by Lerdahl is then used to compute costs between different chord candidates. Dynamic programming is also involved to select the best path among chord candidates. Experiments are performed on a MIDI song database, divided in different music styles. Then the proposed system is compared to the *Melisma Music Analyzer* software proposed by Temperley. Results show that our method has a comparable efficiency and provides not only the root of the chord, but also its mode (major or minor). The proposed system is still open and is able to support more chord types if correct rules to handle them are specified.

1. INTRODUCTION

Harmonic parameters such as keys or chords are essential parameters of the Western music. While the key can be interpreted as the tonal center upon which melody and harmony are built, the chord progression constitutes the harmonic structure of musical pieces. Therefore, chord recognition can be used to figure out the general harmonic progression of a song or a music piece. Such a task is complex, since chords may change in an non-uniform pattern. Another difficulty is that not all the notes of a song are harmonic. Some of them may be ornaments, and thus not belong to the current chord. Possible applications in this field are numerous. Automatic chord recognition can thus be integrated in a music editor software, which would automatically label detected chords. Another possible application may be musical effects such as harmonizers, which could automatically generate an adequate third, fourth or fifth of the note played, based on the background chord detected. Music Information Retrieval (MIR) can also benefit from this field of research, by computing a musical similarity based on chord sequences [2].

Most of the studies in the chord recognition field involve a learning stage, and the use of a hidden Markov model [13], or HMM, especially when dealing with audio. Sheh and Ellis use a 12 dimensional chroma vector feature and the expectation-maximization (EM) algorithm for the training stage [15]. Bello and Pickens adopt the same approach [8], and add some musical information in the model by introducing a transition matrix built on tonal distance considering the circle of fifths (Figure 5). Lee proposes a different feature, the tonal centroid, and does not need an EM algorithm in the training stage [9]. Cabral et al. use a tool that automatically builds an appropriate feature and a adequate learning process depending on the training database [5]. These techniques are detailed and compared in [12]. In all these works, a learning stage is required in order to define the probability of a chord according to a given feature (usually, the chroma vector). They also all use uniform time segmentation.

In this paper, we focus on symbolic music, where music is represented as a list of notes. In the symbolic chord detection literature, some statistical approaches, like Paiement et al.[11] and Rhodes et al.[14] also use training stages. Since the results of such methods depends on the database used for training, we have decided to adopt a different approach. Other systems are rule-based. Rules usually come from musical theory and the technique of dynamic programming may be used for the implementation. As claimed by Temperley [17], the dynamic programming approach indeed provides an efficient way of realizing a preference rule system in a "left-to-right" fashion, so that at each point, the system has a preferred analysis of everything heard so far analogous to the process of real-time listening to music. In 1999, Sleator and Temperley propose the *Melisma Music Analyzer* [16]. *Melisma* first uses a meter analysis process, which gives a chord segmentation of the musical input, before computing a root estimation for each detected chord by using dynamic programming. More recently, Illescas et al. also used dynamic programming to estimate the tonal function of detected chords[7]. This system requires a correct pitch spelling as well as meter information.

Here, we propose a new method which, from a symbolic polyphonic input, estimates a non-uniform chord sequence. As in *Melisma*, each detected chord is labeled with a root. The proposed method also provides a mode (major or mi-

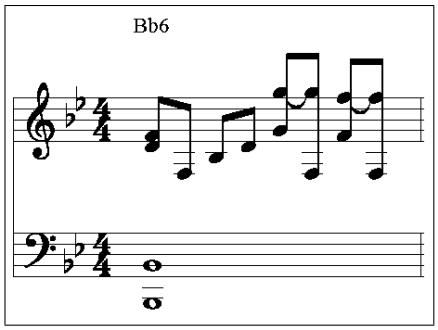

Figure 1. An excerpt of *Bohemian Rhapsody* by Queen. According to a first possible definition, there are 8 different chords in the presented bar (as much as groups of notes sounding at the same time). According to a second possible definition, there is only one single chord in this bar (characterized by the Bb^6 chord label).

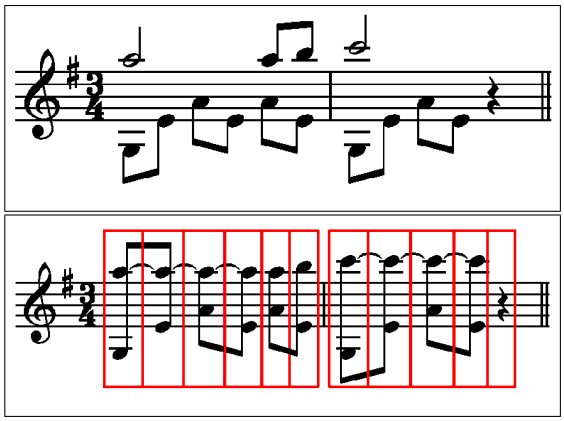

Figure 2. Above: a musical excerpt. Below: the same excerpt with an homorhythmic transformation. Note segments appear bordered in red.

nor). This method relies on dynamic programming to select a sequence among potential chords. Potential keys are also considered, as the transition cost between two chords depends on their respective keys. Neither meter information nor pitch spelling are required. After a discussion on the chord representation is proposed in Section 2, the proposed method is exposed in Section 3. Experiments and results are then presented in Section 4. Finally, Section 5 concludes on the future studies opened by this work.

2. CHORD REPRESENTATION

A chord can be defined in different ways. A chord can simply be defined as the perception of several notes sounding at the same time. However, this first definition is not entirely compatible with the jazz chord notation, where each bar may be labeled with a chord. In this case, the pianist, for example, is not supposed to play a whole note of the specified chord: his only restriction is to play notes or groups of notes belonging *mostly* to the specified chord (some ornaments might be played, which do not belong to the chord). The difference between these two possible definitions of a chord is illustrated in Figure 1. Depending on the definition adopted, it is possible to see either 8 chords or only a single one in this excerpt.

In this paper, we choose the second definition, and we use *chord* to mean the common label a common sequence of notes or groups of notes that could sound at the same time (like the Bb^6 chord label in Figure 1). These consecutive notes or groups of notes within the same chord are called *note segments* of this chord. Each note segment of a given chord may not contain all the notes of the chord, and may contain ornaments (notes which do not belong to the chord). For example, both (C,E,Bb) and (C,G) can be note segments of the C^7 chord. In this paper, the 3 parameters of a chord are:

- the root (the note upon which the chord is built),
- the type (the component intervals defining the chord relatively to the root)
- the mode (major, minor or undefined)

3. DYNAMIC CHORD ANALYSIS

In this section, we first propose a new method for time segmentation by using note segments. The graph of chord candidates is then created, before the dynamic process takes place to select the best path in this graph.

3.1. Time Segmentation

The first issue in the chord detection process is to find an appropriate time segmentation. Audio based methods [5, 9, 15] use time frames in this purpose. It is possible to adopt a similar approach in symbolic music, by using MIDI ticks (or milliseconds) as time units to set the length of an analysis window. We choose a different approach, and assume that a new chord could only occur with a new instance, *i.e.* when at least one new note starts being played, or stops being played. We thus chose to perform an homorhythmic transformation, as introduced in [6], in order to make all the notes sounding at the same time start and end at the same time as well. Therefore no overlapping between notes occurs. This time segmentation defines the different note segments, each of them being formed by notes starting and ending at the same time. Each of these segments potentially starts a new chord. It is important to note that this transformation does not affect the way music is played and heard. An illustration of this homorhythmic transformation can be found in Figure 2. Once this transformation completed, the enumeration of chord candidates can take place for each note segment.

3.2. Graph of Chord Candidates

The note segment constitutes the observation of the dynamic process. A list of hypotheses is built from each observation, these hypotheses being the chord candidates. In the method proposed, a chord candidate is a pair, composed of a compatible chord and a compatible key. Rule-based algorithms are used to determine which chords and keys are compatible with each note segments. The graph of chord candidates is then built: each chord candidate of a given segment is linked to all the candidates of the next segment, thus forming an directed acyclic graph.

3.2.1. Compatible keys

The proposed method enumerates which keys are compatible with each note segment, by using a rule-based approach. Different rules may be used for that purpose. In this paper, we choose to define a key as compatible if each note of the note segment is a component pitch of this key, which means that each note of the segment must belong to the scale of the key (melodic scale for the minor mode). For example, if the note segment is (C,E), the compatible keys are C_{Maj}, F_{Maj}, G_{Maj}, A_{min}, D_{min}, and E_{min}, because both C and E belong to the scales of these keys.

3.2.2. Compatible chords

As for compatible keys, several rules may be defined to determine which chords are compatible with a note segment. We use the following rules, and define different conditions for a chord to be compatible with a note segment, depending on the chord type:

- Maj/min triad chord: such a chord is compatible if each note in the considered segment belongs to the chord. For example, the note segment (C,E) has two compatible triad chords: A_{min} and C_{Maj}.
- Maj/min 7th chord: such a chord is compatible if each note in the considered segment belongs to the chord and if the root and the Maj/min 7th note are present. Such a rule is to avoid having note segment like (E,G,B) or (E,G) be compatible with C^7_{Maj}, for example.
- Maj/min 9th chord: such a chord is compatible if each note in the considered segment belongs to the chord and if the root, the 5th note and the 9th note are present,
- Maj/min 11th chord: such a chord is compatible if each note in the considered segment belongs to the chord and if the root, the 5th note and the 11th note are present.
- other chords may also be compatible, like the min75b chord. For these chords to be compatible, each note is required (for example, a min 75b chord wold be compatible if the considered segment contains the root, the minor 3rd, the flat 5th and the minor 7th).

The proposed system can be modified in order to accept more chord types in the future. Therefore, new rules for any new chord to be compatible with a given note segment must also be specified.

3.2.3. Chord candidates enumerated

The chord candidates finally enumerated are all the possible combination of compatible keys and compatible chords. If n chord and m keys are compatible, n x m pairs are enumerated. For example, with C_{Maj} and A_{min} as compatible chords and C_{Maj} and G_{Maj} as compatible keys, the chord candidates enumerated would be (C_{Maj}, C_{Maj}), (C_{Maj}, G_{Maj}), (A_{min}, C_{Maj}) and (A_{min}, G_{Maj}). If no compatible chord can be enumerated for a given note segment, the hypotheses we choose to set are the previous note segment chord candidates combined with the compatible keys. This is to keep taking into account key detection, even if no chord is compatible. If no compatible key can be built, the hypotheses we choose to set are the same ones as for the previous note segment. Both these choices may be justified by the high probability of having two consecutive note segments being part of the same chord. Finally, the chord output only mentions a root and a mode (major or minor). In other words, even if the type of a chord is supported by our system, only the corresponding mode is taken into account. For example, if a chord is detected as a C^7, it is handled by our system as C_{Maj}. We thus choose to focus on the root and the mode for now, the evaluation of chord types being part of a future work.

3.2.4. Chord transition cost

Once the chord candidates are enumerated for two consecutive note segments, an edge is built from each of the first segment's chord candidates to each of the second segment's. This edge is weighted by a transition cost between the two chord candidates. This transition cost must take into account both the different compatible chords, and the different compatible keys.

We thus choose to use Lerdahl's distance [10] as transition cost. This distance is based on the notion of basic space. Lerdahl defines the basic space of a given chord in a given key as the geometrical superposition of:

a the chromatic pitches of the given key (chromatic level),

b the diatonic pitches of the given key (diatonic level),

c the triad pitches of the given chord (triadic level),

d the root and dominant of the given chord (fifths level),

e the root of the given chord (root level).

Figure 3 shows the basic space of the C_{Maj} chord in the C_{Maj} key. If (C_x, K_x) represents the chord C_x in the key K_x,

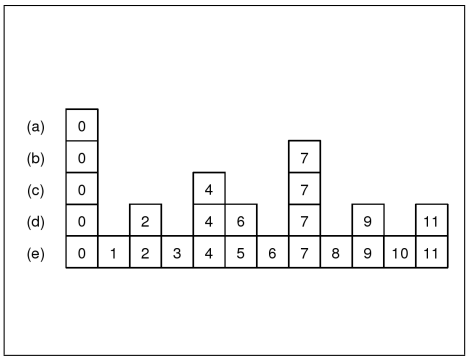

Figure 3. The basic space of the C_{Maj} chord in the C_{Maj} key. Levels (a) to (e) are respectively chromatic, diatonic, triadic, fifths and root levels.

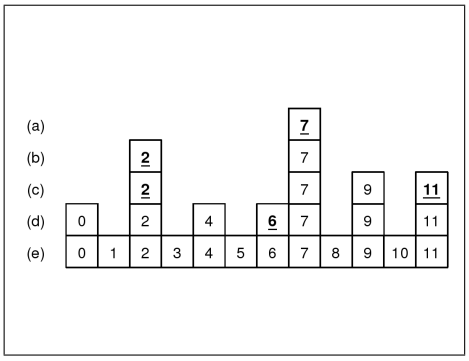

Figure 4. $\delta((C_{Maj},C_{Maj}) \rightarrow (G_{Maj},G_{Maj})) = i + j + k = 1 + 1 + 5 = 7$. The underlined pitches are the non-common pitches.

Lerdahl defines the transition cost from $x = (C_x, K_x)$ to $y = (C_y, K_y)$ as follows:

$$\delta(x \rightarrow y) = i + j + k$$

where i is the distance between K_x and K_y in the circle of fifths (Figure 5), j is the distance between C_x and C_y in the circle of fifths and k is the number of non-common pitch classes in the basic space of y compared to those in the basic space of x.

The distance thus provides a integer cost from 0 to 13, and is completely adequate for a transition cost in the proposed method, since both compatible chords and keys are involved in the cost computation. A calculation of chord transition is illustrated in Figure 4, from $x = (C_{Maj}, C_{Maj})$ to $y = (G_{Maj}, G_{Maj})$. Here, i=j=1 because 1 step is needed to go from C_{Maj} to G_{Maj} in the circle of fifths. k=5 is the number of non-common pitches belonging to the basic space of of y compared to those in the basic space of x (underlined in the Figure). The distance is therefore 1+1+5=7.

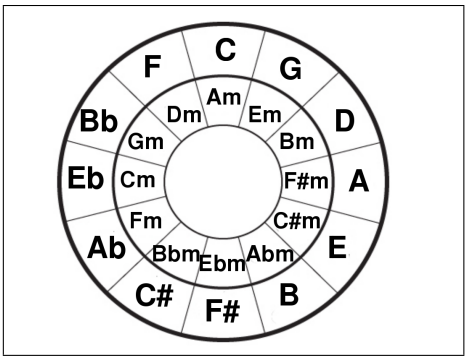

Figure 5. The circle of fifths.

3.3. Dynamic Process

Once the graph between all the chord candidates is formed, the best path has to be found. This task is achieved by dynamic programming [1]. In the graph, from left to right, only one edge to each chord candidate is preserved. Several ways to select this edge can be considered. After experiments, we choose to preserve the edge minimizing the cost to each candidate, as illustrated in Figure 6.c. Other possibilities must be explored in a future work, like preserving the edge minimizing the total sum of costs along the path to the candidate.

The number of final paths is the number of chord candidates for the last note segment. The final path minimizing its total cost sum is then outputted by the program.

3.4. Global Computation

The overall process is illustrated in Figure 6.The construction of the graph is processed in two steps. First, chord candidates for each note segments are determined (Figure 6.a). Costs are then computed between each candidates for the latest note segments, and each chord candidates for the first one (Figure 6.b). After that, the dynamic process takes place: only one edge to a given chord candidate is preserved (Figure 6.c). Finally, the path minimizing its total cost sum is selected (Figure 6.d).

An example of the overall process computation on a musical excerpt is detailed in Figure 7. For each note segment, a chord is computed and the consecutive segments having the same chord identification are then merged, in order to form the boundaries of each chord. In this example, the detected chords are Eb_{Maj} for the first two beats, and Ab_{Maj} for the last two.

4. EXPERIMENTS

In order to perform an evaluation of the proposed method, we gathered a selection of MIDI songs with a specified chord

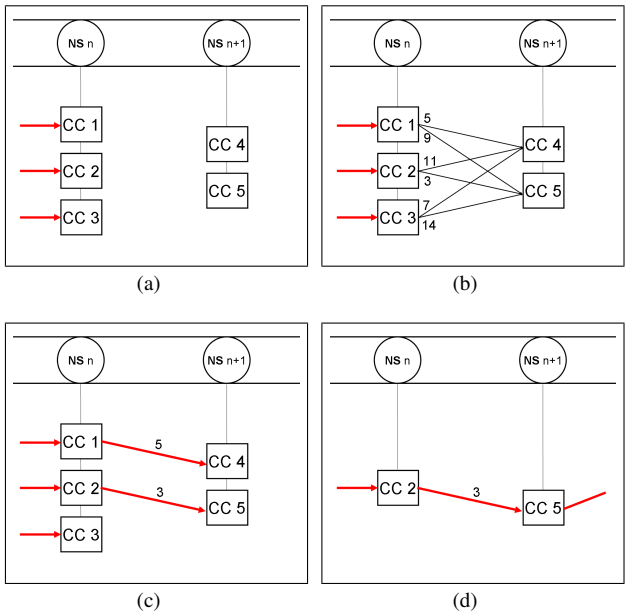

Figure 6. Illustration of the overall process. Graph construction between two consecutive note segments (NS): the possible chord candidates (CC) are listed (**a**), and linked with weighted edges (**b**). Dynamic process: only one edge to each chord candidate is preserved (**c**). In the end, the best path is selected (**d**).

	Songs	Total MIDI ticks (x1000)
latin	37	1177
classical	8	266
pop	45	1866
jazz	65	2405

Table 1. MIDI database used for experiments.

ground truth. We compared the accuracy of our method to the one of *Melisma Music Analyzer*.

4.1. Database

The 155 selected MIDI files are outputs of the software Band-In-A-Box[1]. Band-in-a-Box is a MIDI music arranger software that creates a background for chord progression used in Western Music. The Band-In-Box files we use are taken from Band-In-A-Box Independent Users Group File Archive 2 [3], and are backing tracks. By using these files, we have access to the chord ground truth (specified by Band-In-A-Box) and we are able to generate MIDI backing tracks corresponding to the chord sequences. The songs are already classified in 4 different styles in the archive: *latin, pop, classical* and *jazz*. The database is detailed in Table 1.

[1] http://www.pgmusic.com/products_bb.htm

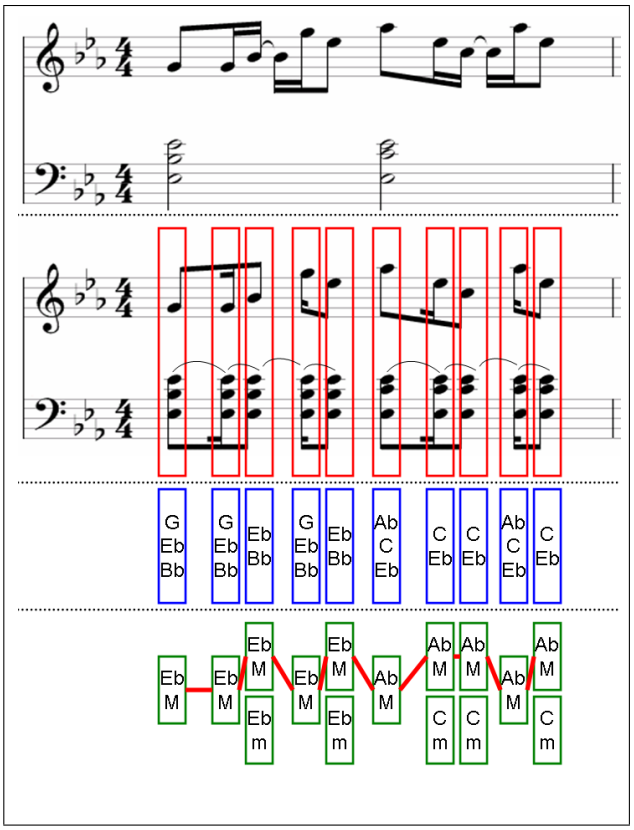

Figure 7. The first bar of *Your Song*, by Elton John, analyzed by our system. Above is the original score. One step below, the score with the homorhythmic transformation. One step below, the note segments. One step below, the chord candidates. For reading purposes, only the chord candidates sharing the same key (here, Eb_{Maj}) are shown. The best path is formed by the chord linked to each others.

4.2. Evaluation Procedure

From audio, chord detection methods are often evaluated frame by frame [8, 9, 15]. For each frame, a chord is estimated, and the global accuracy is the ratio between the number of frames where the estimated chord matches the ground truth and the total number of frames. In the same way, we perform a tick by tick evaluation, by comparing for each MIDI tick the estimated chord and the ground truth, and this for each song of the database. We propose to test the accuracy of both Melisma and the proposed method on this database.

4.2.1. Melisma evaluation

The *Melisma Music Analyzer* is a software developed by Sleator and Temperley [16]. Temperley tested the accuracy of the root extraction of this program on a corpus of excerpts and the 48 fugue subjects from the *Well-Tempered Clavier* by J.S. Bach. The evaluated accuracy on this database is 83.7%.

Since *Melisma* only provides a root for each detected chord, the accuracy on a given song is calculated as the ratio between the number of ticks for which the estimated root matches the ground truth's root and the total number of ticks. To compute a global accuracy for each musical style, two different kind of means were calculated:

- a tick mean, for which a song accuracy is weighted by its number of MIDI ticks,
- a song mean, for which each song accuracy has the same weight

Then, a total accuracy is calculated, by computing the same two means over all the songs from the database.

4.2.2. *System evaluation*

Since our system estimates not only the root, but also the mode (major or minor), it has been decided to perform two different evaluations. The first one is identical to *Melisma*'s evaluation: only the root is taken into account and compared to the ground truth. The second one calculates the accuracy of both root and mode (major or minor) of the estimated chord, according to the following rule: if the chord specified in the ground truth has a mode, it is compared to the estimated chord's mode; but if the chord specified in the ground truth has an undefined mode (*dim* and *sus* chords for example), only the roots are compared, as in *Melisma*'s evaluation. By performing both these evaluations, it is possible to make a direct comparison with *Melisma*'s accuracy.

We also performed a set of evaluations in order to determine which chord types (triads, 7th, 9th, 11th, 13th) to take into account to optimize the system accuracy.

4.3. Results

This section first presents the results of the system handling different chord types. A comparison between the results of the proposed method and *Melisma* on the the root estimation are presented. Finally, results of the system evaluated on the root and the mode of detected chord are exposed.

4.3.1. *Supported chord types optimization*

Extending the number of supported chords increases the number of potential chord candidates for each note segment. By doing so, the probability of having the same two chord candidates in two consecutive segments is also increased. And so is the probability of making the system choose to keep the same chord candidate, even if a change of chord occurs between the two considered note segments (Figure 8). A balance thus needs to be found between extending the number of supported chords and adding errors due to this extension.

The search for optimization in the supported chord types seems to be reached for 11th chords, according to the results presented in Table 2. We note that handling 7th chords has a great effect on the global accuracy. The *latin* and *jazz* styles are especially affected (scores for these two styles being increased by 19% and 16% respectively). This is mostly because those two styles involve a lot of chords more elaborated than triad chords. Handling 9th chords seem only really profitable to these same two musical styles (whose scores gain an extra 4%). Handling 11th chords has a very small effect on the overall scores. Finally, when handling the 13th chords, the overall scores drop notably, especially, once again, on the *latin* and *jazz* styles (scores for these two styles being decreased by 8% and 5% respectively). An explanation could be illustrated by a simple example: when considering the note segment (A,C,E,G), A_{min} is the only compatible chord if supported chords are triads, 7th, 9th and 11th. But when considering 13th chords, (A,C,E,G) can be analyzed as a C^{13}_{Maj} without any 7th, 9th or 11th. Therefore, when handling 13th chords, a note segments like (A,C,E,G) can have both A_{min} and C_{Maj} as compatible chords, which may led to some errors, as illustrated by Figure 8. That is the reason why we choose to limit the system to triads, 7th, 9th and 11th chords as supported chords.

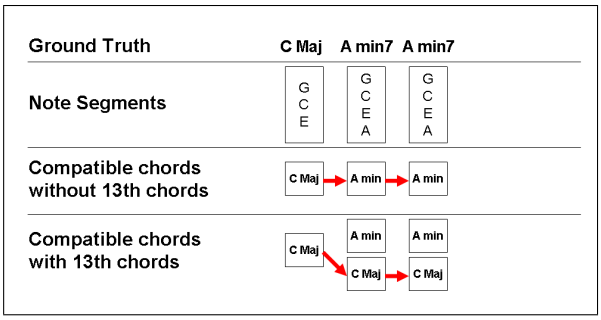

Figure 8. Example of supported chord type extension causing an error. The final chosen path is best (according to the ground truth) when 13th chord are not supported.

4.3.2. *Comparison on the root estimation*

The comparison between the proposed system and *Melisma* can be found in Table 3. If *Melisma* is more accurate on *jazz* songs (according to the tick mean score only), the proposed system gets the best results on *pop, classical* and *latin* songs, as well as *jazz* songs when considering the song mean. The difference is especially noticeable on the *classical* and *latin* styles. On the overall results, our system is very comparable to *Melisma*, the difference being quite small (less than 1% based on the overall MIDI tick mean, and less than 2% based on the overall song mean).

It is also possible to analyze these results in a different way. Figure 9 presents two histograms showing the number of songs depending on the calculated accuracy. The two profiles are close, but not identical. *Melisma* smallest score for a given song is 41.9%, when our system's is 36.5%. The highest score reached for the two systems is

	Triads (5th)		5th+7th		5th+7th+9th		5th+7th+9+11th		5th+7th+9th+11th+13th	
	T.M.	S.M.	T.M.	S.M.	T.M.	S.M.	T.M.	S.M.	T.M.	S.M.
latin	60.6	66.4	79.4	82.4	83.6	86.8	**84.1**	**87.4**	76.4	82.5
classical	71.7	72.4	86.5	86.0	**87.0**	**86.4**	86.8	86.3	85.0	85.6
pop	79.7	76.2	86.1	85.2	87.2	86.8	**87.4**	**87.2**	85.4	84.0
jazz	53.4	52.4	69.0	68.2	72.9	72.0	**73.1**	**72.3**	67.9	66.0
Total	63.3	63.7	77.6	77.4	80.4	80.6	**80.7**	**80.9**	76.2	76.2

Table 2. Accuracy (evaluated for the chord root, and shown in %) depending on the supported chord types. T.M. stands for Tick Mean, and S.M. for Song Mean. The best accuracy is reached for triads, 7th, 9th and 11th chords.

	Our System		Melisma	
	T.M.	S.M.	T.M.	S.M.
latin	**87.0**	**89.1**	85.2	86.1
classical	**86.1**	**85.5**	80.7	80.3
pop	**89.1**	**89.5**	88.5	88.8
jazz	76.9	**77.3**	**77.6**	75.7
Total	**83.4**	**84.1**	82.9	82.2

Table 3. Results on the Band-In-A-Box database. The accuracy is in %. T.M. stands for Tick Mean, and S.M. stands for Song Mean. Only the root of the chords are evaluated. Our system performs a comparable accuracy to *Melisma*.

100%, but for different songs. Out of 155 songs, our system gets 102 songs above 80% accuracy when *Melisma* gets 92 songs. More remarkable, the 3 songs maximizing the score difference between the two systems are all better analyzed by our system: *April Joy* (90% to 51.2%), *A Taste Of Honey* (86.4% to 42.1%) and *The Entertainer* (95.5% to 41.2 %). It is also interesting to note that these 3 songs have been categorized in 3 different musical styles. On the other hand, *Melisma* sometimes gets better results, for example on *Alone Together* (84.2% to 63.7%) and *April Showers* (95.3% to 75.6 %).

Such statistics show that even if the two compared methods get comparable scores, they seem to use different kinds of information. The use of a meter analysis in *Melisma*, as well as its handling of ornamental notes are two major differences between the two systems that may explain why the two methods get different results.

4.3.3. Root and mode estimation

Table 4 presents the results of our system when evaluated on the root and the mode of the detected chords. In the *pop, latin*, and *classical* databases, the results are slightly better than *Melisma* evaluated on the root only (83.3%). The overall result calculated with a song mean is also slightly better than *Melisma*. Our system is thus very comparable to *Melisma* when it comes to accuracy, and provides not only a root, but also a mode for each detected chord.

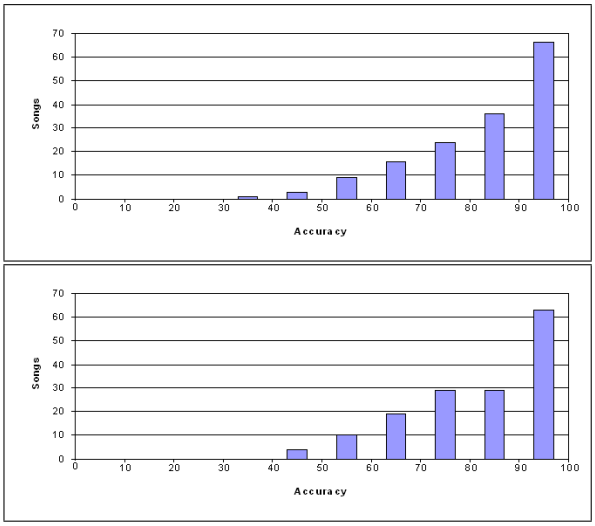

Figure 9. Histograms representing the number of files per interval of evaluated accuracy for the proposed system (above) and *Melisma* (below). The two systems seem to carry different kind of information.

	Our System	
	Tick Mean	Song Mean
latin	86.8	88.9
classical	84.2	83.2
pop	88.8	89.1
jazz	75.3	76.0
Total	82.5	83.3

Table 4. Results on the Band-In-A-Box database. The accuracy is in %, and both the root and the mode of the chords are evaluated. The system shows a comparable accuracy to *Melisma* evaluated on the root only.

5. CONCLUSION AND FUTURE WORK

We propose a new method for chord estimation from symbolic polyphonic data. This method involves non-uniform time segmentation, dynamic programming, Lerdahl's chord transition cost and rule-based chord candidates enumeration for each note segment. Experiments have been performed in order to evaluate the accuracy of the proposed method. A comparison has been proposed between the proposed system and *Melisma* Music Analyzer. Results show that the proposed method has a slightly better accuracy than *Melisma* when evaluated on the root only. When evaluated on the root and the mode of the detected chords, the accuracy is comparable to the root detection accuracy of *Melisma*.

These results seem promising, especially when the proposed system may be extended, by handling a larger number of supported chord types, taking into account a larger set of rules, choosing a different way to select the best path in the graph of chord candidates or using a different distance for chord transition cost. Detection of key change, as studied in [4] is also planned, since compatible keys are determined for each note segment. A large database of symbolic music with key change annotations will be needed for evaluation. Evaluating the type of chord is also possible, since each chord could be handled by the system with a given type. Possible applications to Music Information Retrieval (MIR) and adaptation to audio chord recognition are under progress.

6. ACKNOWLEDGMENT

This work is part of the SIMBALS project (JC07-188930), funded by the French National Research Agency (ANR), and is also supported by the Aquitaine Regional Council.

7. REFERENCES

[1] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[2] J. Bello, "Audio-based Cover Song Retrieval using Approximate Chord Sequences: Testing Shifts, Gaps, Swaps and Beats," in *Proc. of the 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007, pp. 239–244.

[3] A. M. Birch and O. Anon, *Band-In-A-Box Independent Users Group File Archive 2*, Available: http://groups.yahoo.com/group/Band-in-a-Box-Files2/, 2004.

[4] E. Chew, "The Spiral Array: An Algorithm for Determining Key Boundaries." in *Proc. of the Second International Conference ICMAI 2002*, Springer, 2002, pp. 18–31.

[5] G. Cabral and F. Pachet and J. Briot, "Automatic X Traditional Descriptor Extraction: The Case of Chord Recognition," in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, U.K., 2005, pp. 444–449.

[6] P. Hanna, M. Robine, P. Ferraro, and J. Allali, "Improvements of Alignment Algorithms for Polyphonic Music Retrieval," in *Proc. of the CMMR08, International Symposium on Computer Music Modeling and Retrieval*, Copenhagen, Denmark, 2008, pp. 244–251.

[7] P. Illescas, D. Rizo, and J. M.Iesta., "Harmonic, Melodic, and Functional Automatic Analysis," in *Proc. of the International Computer Music Conference (ICMC)*, Copenhagen, Denmark, 2007, pp. 165–168.

[8] J. P. Bello and J. Pickens, "A Robust Mid-Level Representation for Harmonic Content in Music Signals," in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, U.K., 2005, pp. 304–311.

[9] K. Lee and M. Stanley, "Acoustic Chord Transcription and Key Extraction from Audio Using Key-Dependent HMMs Trained on Synthesized Audio," *The IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 2, pp. 291–301, 2008.

[10] F. Lerdahl, *Tonal Pitch Space*. Oxford University Press, 2001.

[11] J.-F. Paiement, D. Eck, and S. Bengio, "A Probabilistic Model for Chord Progressions," in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, UK, 2005, pp. 312–319.

[12] H. Papadopoulos and G. Peeters, "Large-scale Study of Chord Estimation Algorithms Based on Chroma Representation and HMM," in *Proc. of the 5th International Conference on Content-Based Multimedia Indexing*, Bordeaux, France, 2007, pp. 53–60.

[13] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Readings in speech recognition*, 1990, pp. 267–296.

[14] C. Rhodes, D. Lewis, and D. M$\acute{}$ullensiefen, "Bayesian Model Selection for Harmonic Labelling," *Mathematics and Computation in Music*, 2007.

[15] A. Sheh and D. P. W. Ellis, "Chord Segmentation and Recognition Using EM-Trained Hidden Markov Models," in *Proc. of the 4th International Conference on Music Information Retrieval (ISMIR)*, Baltimore, U.S.A., 2003, pp. 183–189.

[16] D. Temperley and D. Sleator, "The Melisma Music Analyzer," http://www.link.cs.cmu.edu/music-analysis/.

[17] D. Temperley, *The Cognition of Basic Musical Structures*. The MIT Press, 1999.