Journal of New Music Research
2005, Vol. 34, No. 4, pp. 375–393

Multi-Dimensional Motivic Pattern Extraction Founded on Adaptive Redundancy Filtering

Olivier Lartillot

University of Jyväskylä, Finland

Abstract

We present a computational model for discovering repeated patterns in symbolic representations of monodic music. Patterns are discovered through an incremental adaptive identification along a multi-dimensional parametric space. The difficulties of pattern discovery mainly come from combinatorial redundancies, that our model is able to control efficiently. A specificity relation is defined between pattern descriptions, unifying suffix and inclusion relations and enabling a filtering of redundant descriptions. Combinatorial proliferation caused by successive repetitions of patterns is managed using cyclic patterns. The modelling of these redundancy control mechanisms enables an automation of musicology-relevant analyses of musical databases.

1. Introduction

1.1 Solving a core musicological issue

Music is a domain of expression that features a significant amount of structural complexity: musical pieces contain a huge number of notes and structures that are composed in a more or less conscious way, and that listeners perceive only partially. One objective of music analysis, and in particular of motivic analysis, is to explicitly show these interesting structures hidden in musical scores. The task, however, remains highly difficult. During the twentieth century, improvements of traditional motivic analysis have been attempted (Reti, 1951) but faced up to two major difficulties. Firstly, for all the tremendous amount of energy dedicated to the deep analysis of musical pieces, complete and exhaustive analyses of complex pieces could hardly been achieved. Secondly, the objective relevance of these analyses was not assured, as most of the analytic processes were carried out intuitively (Cook, 1987). In the second half of the twentieth centuries, researches have tried to improve the objectivity of motivic analysis through a formalisation of the discovery processes (Ruwet, 1987; Nattiez, 1990). However, the formalisations have not been justified neither actually applied on practical examples.[1]

Nowadays, computer may help solving the two difficulties described before, namely, the management of music complexity and the control of objectivity. Indeed, the automation of analysis processes would enable a thorough and exhaustive study of musical pieces of any length. The automation requires a fully explicit description of discovery processes, which might ensure in return the objectivity of these analyses.

The trouble is, up to now, computational automations of motivic analysis fell short of musicologists' expectations. One reason for this failure is the existence of a problem of combinatorial redundancy, which comes from the definition of the pattern discovery task. We assume that this redundancy is implicitly controlled by the cognitive system founding listening and analysis

[1]The practical application of Ruwet's method (Ruwet, 1987) is in fact founded on the author's own implicit intuitions. Indeed, an accurate computer running of the proposed formalisation would lead to absurd results (Lartillot, 2004b). This shows in particular the limitations of structuralist approaches of music that refuse to take into account listening strategies (Lartillot & Saint-James, 2004).

Correspondence: Olivier Lartillot, University of Jyväskylä, Department of Music, PL 35(A), 40014 University of Jyväskylä, Finland. E-mail: lartillo@campus.jyu.fi

DOI: 10.1080/09298210600578246 © 2005 Taylor & Francis

processes. A cognitive modelling of motivic analysis is therefore proposed that includes mechanisms of redundancy control and that offers results of good quality.

1.2 Culture-independent musical patterns

Current researches in automated motivic analysis – including ours – do not generally study the impact of cultural knowledge on pattern discovery, despite its significance in actual listening activities. Indeed, the integration of cultural strategies into the computational model is far from evident, and raise intricate questions. It seems more convenient to focus first on non-cultural processes, and to integrate cultural strategies afterwards.

Non-cultural pattern discovery is generally carried out following two distinct strategies. On the one hand, temporal gaps and musical discontinuities (such as pitch leaps, or changes in intensity, timbre, etc.) induce the determination of boundaries (Lerdahl & Jackendoff, 1983; Cambouropoulos, 2004). On the other hand, listeners, experienced or not, easily perceive repeated patterns, representing therefore one of the most salient characteristics of musical works, sometimes called *parallelism* (Lerdahl & Jackendoff, 1983; Ruwet, 1987). Our approach is currently focused on this second strategy of repetition discovery.

1.3 Related works

Several approaches have been proposed to the problem of repeated pattern extraction in symbolic representations of music. Cambouropoulos (2004) looks for exact pattern repetition, using Crochemore's (1981) approach, in different parametric descriptions of musical sequences. As the set of patterns produced by the algorithm is not highly relevant, an estimation of the segmentation points is computed through a weighted average of the segmentations implied by each different pattern.

In Conklin & Anagnostopoulou (2001), pattern discovery is performed by building a suffix tree data structure along several parametric dimensions. Once again, as the set of discovered patterns is large and poorly relevant, a subsequent step select patterns that occur in a specified minimum number of pieces and that satisfy a statistical significance criterion. A further filtering globally selects the longest significant patterns within the set of discovered patterns.

Contrary to strict monodic approaches focused on the discovery of repeated subsequences – formed by notes immediately successive in the original sequence – other approaches adopt a more general definition of patterns, where successive notes do not need to be immediately successive in the original sequence. Rolland (1999) defines a numerical similarity distance between subsequences based on edit distance. In order to extract patterns, similarity distances are computed between all possible couples of subsequences of a certain range of lengths, and only similarity exceeding a user-defined arbitrary threshold are selected. From the resulting similarity graph, patterns are extracted using a categorisation algorithm called *Star Center*. The set of discovered patterns is reduced even further using offline filtering heuristics. In particular, only patterns repeated in a minimum number of musical sequences are selected.

Meredith, Lemström & Wiggins (2002) generalise the pattern extraction task to polyphony. Notes of musical sequences are represented by points in a two-dimensional (pitch/time) dimension, and maximal repetitions of point sets are searched. This geometrical strategy does however not apply to melodic repetitions that present rhythmic variations. Post-processing techniques have been added that perform global selection in order to enhance the precision factor.

1.4 Problem statement

In all the approaches presented in the previous section, the initial pattern discovery algorithm produces a huge amount of motives that do not generally correspond to actually perceived structures. The complexity is commonly reduced through a filtering of the results using statistical criteria. However, this filtering does not improve the perceptive relevance of the results, and may arbitrarily discard interesting patterns. Our approach consists in making explicit the different factors responsible for the bad behaviour of common pattern extraction algorithms. A better control of all these factors enables a robust extraction of relevant patterns, and therefore a detailed analysis exempt from global averaging or filtering. The problem can actually be decomposed into several different distinct general subproblems. They will be presented in the reminder of the paper, with mechanisms solving each subproblem as simply as possible.

2. Basic framework

This section introduces and justifies the basic concepts and definitions that are used throughout the paper, such as the musical parameters, the identification strategy and the knowledge representation.

2.1 The musical dimensions

2.1.1 Pitch dimensions

Music is expressed along multiple parametric dimensions (Figure 1). *Theoretical pitch values*, such as C#, stem from the existence of tonal or modal *scales*. Each theoretical pitch value may hence be expressed as a *degree* on this scale. This scale degree can be represented on a numerical

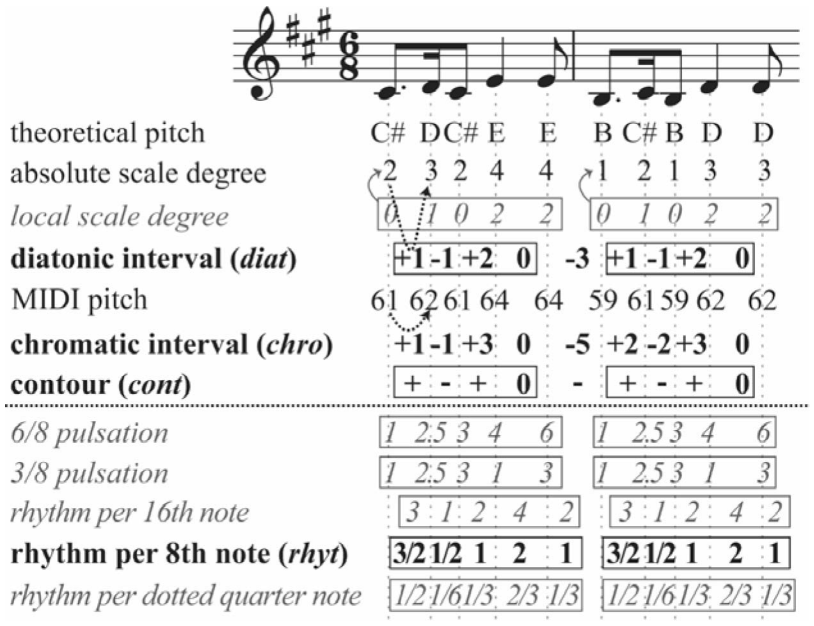
theoretical pitch C# D C# E E B C# B D D
absolute scale degree 2 3 2 4 4 1 2 1 3 3
local scale degree 0 1 0 2 2 0 1 0 2 2
diatonic interval (diat) +1 -1 +2 0 -3 +1 -1 +2 0
MIDI pitch 61 62 61 64 64 59 61 59 62 62
chromatic interval (chro) +1 -1 +3 0 -5 +2 -2 +3 0
contour (cont) + - + 0 - + - + 0
6/8 pulsation 1 2.5 3 4 6 1 2.5 3 4 6
3/8 pulsation 1 2.5 3 1 3 1 2.5 3 1 3
rhythm per 16th note 3 1 2 4 2 3 1 2 4 2
rhythm per 8th note (rhyt) 3/2 1/2 1 2 1 3/2 1/2 1 2 1
rhythm per dotted quarter note 1/2 1/6 1/3 2/3 1/3 1/2 1/6 1/3 2/3 1/3

Fig. 1. Description of a musical sequence following different musical dimensions. Repeated sequences of values, which form patterns, are squared. Are highlighted the dimensions integrated in our approach. Dimensions in grey will be considered in future works.

scale centred on the tonic of the scale (cf. *absolute scale degree*, Figure 1).

Alternatively, the pitch of each note may be expressed independently of any scale. Particularly convenient for that purpose is the *chromatic pitch* representation, which associates with each enharmonic pitch – say, each key of a piano keyboard – a position number. Following the MIDI standard, with C4 is associated the value 60, and the pitch value of each other note is computed with respect to the pitch distance in semi-tones from C4 (cf. *MIDI pitch*, Figure 1).

Pitches may sometimes be perceived in an absolute way by expert listeners, but the melodic dimensions of patterns is mainly considered as a relative configurations determined by context. One interesting approach to pitch local configuration consists in representing pitches as 'abstracted chroma' (Dowling & Harwood, 1986, p. 128): When a motive is transposed (as in Figure 1), listeners actually perceive the repetition of a same pattern of pitch height (0,1,0,2,2, in the *local scale degree* dimension of Figure 1), but only the local scale has changed, rooted from one scale degree to another. This more detailed modelling will be considered in future works.

Following a long tradition in computer music researches, current version of the proposed modelling is based on a simpler strategy, which consists in defining pitches of each note with respect to its previous one, forming an interval called *inter-pitch*. In the diatonic dimension, inter-pitch, called *diatonic interval* and noted '*diat*' in the remainder of the paper, represents scale degree differences between successive notes. Similarly, chromatic intervals, noted '*chro*', represents chromatic pitch differences between successive notes.

In this way, transposed patterns correspond to explicit repetitions along relative dimensions, be they modelled using inter-pitches or local scales. Absolute pitch information can then be used in order to specify patterns: amongst transposed occurrences of same patterns, occurrences that contain same absolute pitch representation form more specific[2] patterns.

Contour – denoted '*cont*' – simply represents the sense of variation between successive notes: increasing (+), decreasing (−), or constant (0).

2.1.2 Time dimensions

As for pitches, absolute descriptions of temporal positions seem to have a limited impact on pattern perception. Temporal positions are usually considered with respect to the implicit metrical structure, which consists of a hierarchy of pulsation levels. A 6/8 metre, for instance, define four principle pulsation levels synchronised respectively with bars, dotted quarter notes, 8th notes and 16th notes. Metric positions can then be formalised by cycles with transitions synchronised with one pulsation level, and with a period synchronised with a higher pulsation level. For instance, in Figure 1, a 6/8 pulsation can be formalised using a 6-state cycle with 8th-note transitions and bar period. Similarly, a 3/8 pulsation may be defined as a 3-state cycle with periods beginning at each upbeat,

[2]The notion of specificity will be further developed in Section 3.1.

i.e. the first and fourth pulsation of each bar. These pulsation dimensions enable the detection of class of motives starting on an up-beat, a down-beat, or a particular metrical position, and eventually reject patterns that do not start on the same beat accent or metrical position and that cannot be perceived for this reason. The integration of this filter in the modelling will be considered in future works.

A simpler strategy, followed by most computational approaches – including ours – is based on relative configurations, such as note durations or temporal distances between successive note-onset positions (or *inter-onset*). A direct measurement in seconds of these distances is insufficient, as it does not take into account the variation of tempos. Inter-onsets are preferably defined by dividing the temporal distance by the metrical pulsation. The multiplicity of metrical pulsations, as explained in previous section, leads to a multiplicity of rhythmical dimensions. Figure 1, for instance, displays three rhythmic dimensions: rhythm per 16th note, 8th note and dotted quarter note. Augmented and diminished occurrences can be detected through a global search among all these pulsation levels together. The current version of the modelling is restricted to one single rhythmic dimension, denoted '*rhyt*' in the article, and based on the metrical unit defined by the bottom number of the time signature (8th note for a 6/8 metre).

Another way to handle relative rhythmic configurations is based on the definition of a quotient, called *inter-onset ratio*, between successive inter-onsets (for instance Rolland, 1999; Conklin and Anagnostopoulou, 2006; Lartillot, 2004a). The parameter of inter-onset ratio is however insufficiently intuitive and cannot be generalised to polyphony neither to complex monodies, where multiple intervals can be drawn from one single note, therefore implying multiple possible inter-onset ratio values for a same note.

Current version of the modelling uses in total four relative parameters – diatonic interval, chromatic interval, contour and rhythmic value – that describe each interval between successive notes. Here, 'interval' is considered in a general sense: even the rhythmic value related to one note can be considered as a temporal interval between that note and its successive one.

2.2 An adaptive pattern identification

Patterns are generally not exactly repeated but transformed in multiple ways. These approximate repetitions should therefore be detected despite the superficial differences. Current approaches follow two different strategies. One is based on numerical similarity (Figure 2, top): patterns are identified when dissimilarity does not exceed a given threshold (Cope, 1991; Rolland, 1999; Cambouropoulos et al., 2002). This threshold cannot however be defined precisely, and is therefore tuned arbitrarily by the user, which questions the general relevance of the results based on this strategy. On the other hand, reference cognitive studies (Dowling & Harwood, 1986) assume that pattern identifications do not come from numerical distance minimisation, but rather from exact identification along multiple musical dimensions of various specificity levels, such as pitch, contour and rhythm (Figure 2, bottom).

Several approaches to pattern discovery follow this second strategy of identification along multiple musical dimensions (Cambouropoulos & Tsougras, 2004; Conklin & Anagnostopoulou, 2001; Meredith et al., 2002) and search for repetitions along each different dimension and product of dimensions. Nonetheless, there exist patterns that are progressively constructed along variable successive musical dimensions. These *heterogeneous* patterns cannot be identified using traditional approaches. For instance, each line of the score in Figure 3 repeats a same pattern: in the first half, both melodic and rhythmic descriptions are repeated whereas, in the second half, only the rhythmic descriptions are repeated. In order to discovery heterogeneous patterns, all the musical dimensions have to be considered at each phase of the progressive construction, and relevant viewpoints have to be selected in an adaptive way. A computational solution to this core problem is described in this paper.

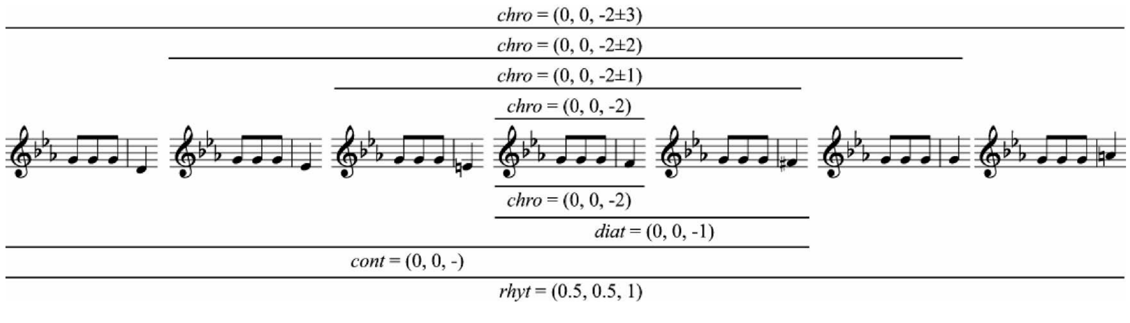

Fig. 2. Two pattern classification methods: one (top) based on a numerical similarity distance, for instance, along the chromatic dimension *chro*, the other (bottom) based on exact identification along different musical dimensions.

2.3 Graph-based data representation

2.3.1 Pattern chains

Patterns are formalised as chains of states – called *pattern chains* (PCs) – that represent the succession of prefixes, corresponding to the successive steps of the incremental pattern construction. For instance, in Figure 4, a pattern of five notes is repeated two times. The chain of states $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$ represents the successive extensions of the pattern, where the final state *e* corresponds to the whole pattern. The successive transitions between states contain the descriptions of the successive intervals forming the pattern. Similarly, each pattern occurrence is represented as a chain of states – called *pattern occurrence chain* (POC) – featuring the successive prefixes too. Each state of POCs are related to each respective state of the corresponding PCs.

2.3.2 Pattern trees

As patterns can accept multiple alternative continuations, PCs can be extended by multiple branches. Hence, patterns are aggregated into one single tree, called *pattern tree* (PT), and each PC is as a branch of the PT (see Figure 5 for instance). Similarly, each pattern occurrence can accept multiple alternative continuations. Hence, the set of all pattern occurrences that are initiated by one note forms a tree, called *pattern occurrence tree* (POT) (see Figure 7 for an example). The root of each POT is related to the root of the PT (node *a* in Figure 7), which represents the simple concept of note, and is therefore called *note pattern*. Since all notes can potentially initiate a POT, they are all occurrences of the *note pattern*.

3. Reducing the combinatorial redundancy

A running of the basic algorithm on musical examples, even simple, produces a huge number of patterns that do not correspond, for most of them, to actually perceived structures, entailing significant combinatorial explosion. This is due first to the fact that our approach accepts a large number of possible configurations, such as heterogeneous patterns. Yet, all computational approaches face up to similar combinatorial problems, as shown in Section 1.3. In fact, the repetition-based paradigm implicitly leads to this kind of redundancy. We propose to decompose the problem into several distinct subproblems, and to introduce mechanisms that resolve each of them as simply as possible.

3.1 A restriction to closed patterns

Combinatorial explosion is a direct consequence of the pattern discovery task itself. Indeed when a pattern of

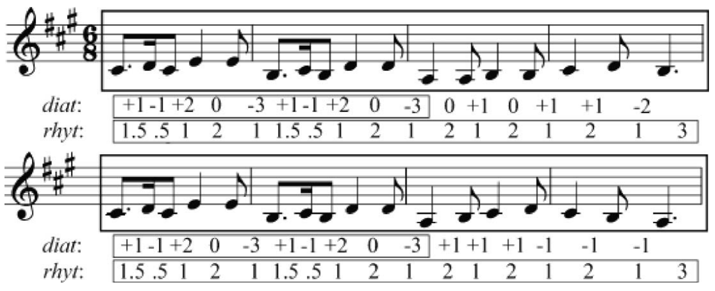
diat: +1 -1 +2 0 -3 +1 -1 +2 0 -3 0 +1 0 +1 +1 -2
rhyt: 1.5 .5 1 2 1 1.5 .5 1 2 1 2 1 2 1 2 1 3
diat: +1 -1 +2 0 -3 +1 -1 +2 0 -3 +1 +1 +1 -1 -1 -1
rhyt: 1.5 .5 1 2 1 1.5 .5 1 2 1 2 1 2 1 2 1 3

Fig. 3. Repetition of a heterogeneous pattern (squared): the first half is melodico-rhythmic (melodic and rhythmic descriptions are repeated) and the second half is simply rhythmic (only the rhythmic descriptions are repeated).

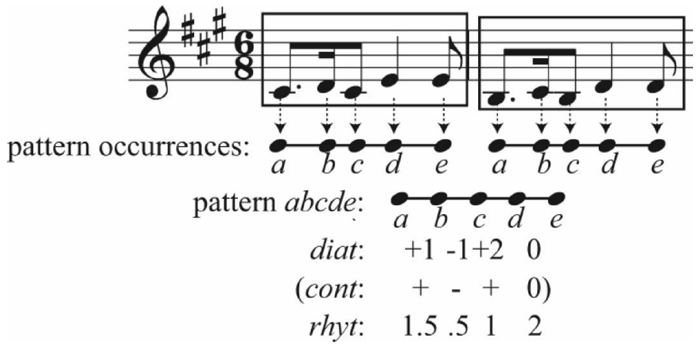

Fig. 4. *Pattern occurrences* are formalised as chains of states, each successive state representing each successive note of each occurrence. The *pattern* itself is represented by the same chain of states. Its description is represented along the transitions between its successive states. Here, the contour representation, redundant with the diatonic representation, can be discarded.

length l is discovered, all its $2l-1$ subpatterns (which are either prefixes, suffixes or suffixes of prefixes of the initial pattern) may be considered as patterns too and would then be discovered explicitly by any brute-force algorithm. One way to avoid this redundancy, as considered for instance by Meredith, Lemström and Wiggins (2002), consists in focusing only on *maximal patterns*, that is: patterns that are not subpatterns of other patterns. This heuristics enables a significant reduction of redundancy, but leads also to an important loss of information. For instance, pattern *j* in Figure 5 is a simple suffix of pattern *e*. It does not need to be explicitly represented, since the set of its occurrences (or *pattern class*) can be directly deduced from the class of its superpattern *e*.

In Figure 6, on the contrary, the pattern class of *j* cannot be directly deduced from the pattern class of *e*, and should therefore be explicitly represented in the final analysis. This principle corresponds exactly to the notion of *close patterns*, which are patterns whose number of occurrences (or *support*) is not equal to the support of their superpatterns.

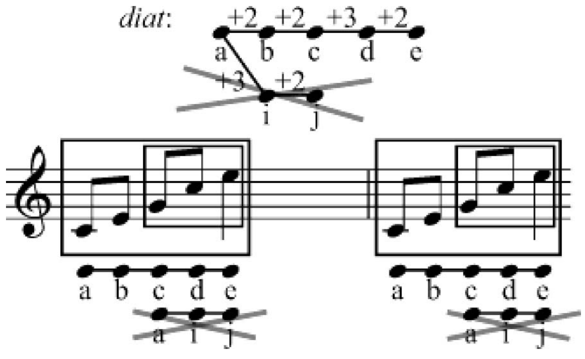

Fig. 5. Pattern *j* is a simple suffix of pattern *e*, and both pattern classes are equal. Therefore pattern *j* should not be explicitly represented.

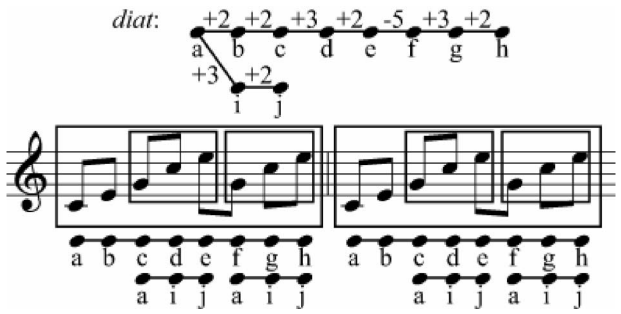

Fig. 6. Pattern *j* is not a simple suffix, of either *e* or *h*, since their classes are strictly included in the class of pattern *j*. Pattern *j* should therefore be explicitly represented.

The model presented in this paper looks for closed pattern in musical sequences. For this purpose, the notion of inclusion relation between patterns founding the definition of closed patterns needs to be generalised to the multi-dimensional parametric space of music, defined in previous section. This problem can be solved using the Gallois correspondence between pattern classes and pattern description, as studied in Formal Concept Analysis theory in particular (Ganter & Wille, 1999): each pattern may be considered as a concept $C=(G, M)$, where G is the pattern class, or set of objects of the concept, and M is the pattern description. A notion of subconcept-superconcept relation between concepts is defined: $C_1=(G_1, M_1)$ is a subconcept and $C_2=(G_2, M_2)$ is the corresponding superconcept, if the description M_1 is included into the description of M_2, or equivalently if the pattern class G_2 is included into the pattern class G_1 (Zaki, 2005). A subconcept is considered as *less specific* than its superconcept.

For instance, pattern *e* (in Figure 7) features melodic and rhythmical descriptions, whereas pattern *i* only features the rhythmic part. Hence pattern *e* can be considered as more specific than pattern *i*, since its description contains more information.

The general principle ruling the pattern redundancy control consists in a multi-dimensional closed pattern discovery using the specificity relation, which may be stated as follows: If a pattern *i* is less specific than another pattern *e*, and if, in the same time, the pattern class of *i* is equal to the pattern class of *e*, then the pattern *i*, considered as *non-closed* and therefore *redundant*, should not be inferred at all. Or to put it another way, among the multiple descriptions of a pattern class, only the most specific one should be explicitly considered, since it implicitly contains the less specific descriptions. In Figure 7, on the contrary, the less specific pattern *i* is closed and can be explicitly represented since the third occurrence is not an occurrence of the more specific pattern *e*.

3.2 Periodic sequences

Combinatory explosion can be caused by another common phenomenon provoked by successive repetitions of a same pattern (for instance the simple rhythmic pattern *c*, a succession of 8th note and quarter note, in Figure 8). As each occurrence is followed by the beginning of a new occurrence, each pattern can be extended (leading to pattern *d*) by a new interval whose description (a 8th note) is identical to the description of the first interval of the same pattern (i.e. pattern *b*). This extension can be prolonged recursively (into *e*, *f*, etc.), leading to a combinatorial explosion of patterns that are not perceived due to their complex intertwining (Cambouropoulos, 1998).

3.2.1 Related work

This phenomenon of successive repetition, although very frequent in musical expression, has been rarely studied (Cambouropoulos, 1998; Conklin & Anagnostopoulou, 2006). In Cambouropoulos (1998), for instance, the combinatorial explosion generated by the phenomenon is reduced by selecting, once the analysis is completed, the patterns featuring minimal temporal overlapping between occurrences. The global degree of overlapping is defined as the fraction of overlapping notes among the notes of the pattern class. However, only one aspect of the redundancy problem is taken into consideration here. For instance, the first and last occurrences of pattern *e* (Figure 8) do not overlap but should however be filtered. Moreover, as the selection is globally inferred, interesting and relevant patterns may be discarded. The management of periodicity-based proliferation requires a close examination of each specific occurrence rather than an examination of the global pattern class. Besides combinatorial redundancy remained problematic since the filtering was done after the actual pattern discovery phase.

3.2.2 Cyclic patterns

The graph-based representation (Figure 8) shows that the last state of each occurrence of pattern *c* is synchronised with the first state of the following occurrence. Listeners seem to tend to fusion these two states, and to perceive a

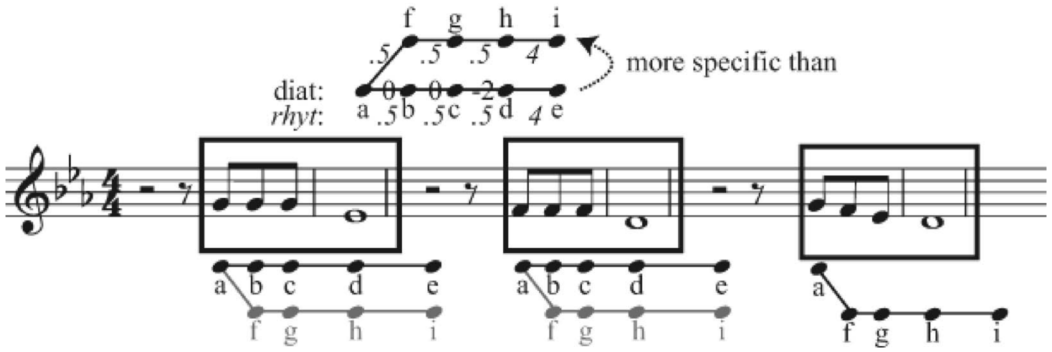

Fig. 7. After analyzing the four first bars, both patterns *e* and *i* having same classes, only the more specific pattern *e* should be explicitly represented. The less specific pattern *i* will be represented once the third occurrence is discovered, as it is not an occurrence of the more specific pattern *e*.

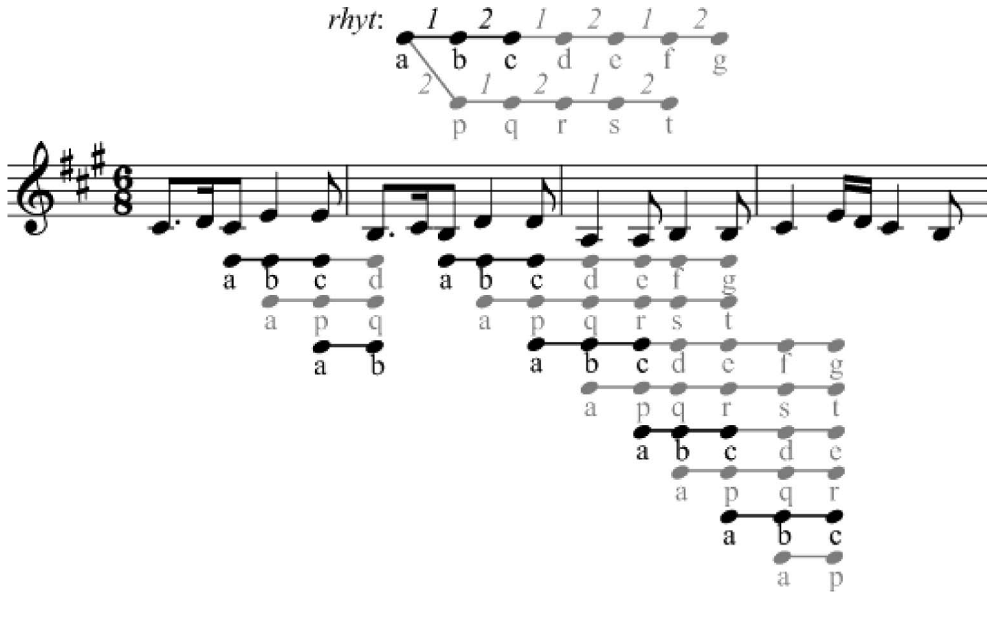

Fig. 8. Multiple successive repetitions of pattern *c* logically lead to extensions into patterns *d*, *e*, etc. and suffixes *p*, *q*, etc., which form a complex intertwining of non-perceived structures.

loop from the last state (c) to the first state (a) (Figure 9). The initial acyclic pattern c leads therefore to a cyclic pattern that oscillates between two states b' and c', corresponding to rhythmic values 1 and 2. Indeed, when listening to the remainder of the musical sequence, we actually perceive this progressive oscillation between these two rhythmic values. Hence this cycle-based modelling seems to explain a common listening strategy, and resolve the problem of combinatorial redundancy.

This cyclic PC (between the two states b' and c' at the top of Figure 9) is considered as a continuation of the original acyclic PC $a \to b \to c$. Indeed, the first repetition of the rhythmic period is not perceived as a period as such but rather as a simple pattern: its successive notes are simply linked to the progressive states a, b and c of the acyclic PC. On the contrary, the following notes extend the POC, which cannot therefore be associated with the acyclic PC anymore, and are therefore linked to the successive states of the cyclic PC (b' and c'). The whole periodic sequence constitutes therefore a single POC representing the traversal of the acyclic PC followed by the cyclic PC.

It can be remarked also that, by property of the cyclic PC, no segmentation is explicitly represented between successive repetitions. For instance, in Figure 9, listeners may perceive a rhythmic period composed of a succession of a 8th note and a quarter note, or on the contrary a succession of a quarter note and a 8th note. Indeed, the listener may be inclined to segment at any phase of the cyclic PC (or not to segment at all).

3.2.3 A necessary chronological pass

This additional concept immediately solves the redundancy problem. Indeed, each type of redundant structure considered previously is a non-closed suffix of prefix of the long and unique POC, and will therefore not be represented any more. But this compact representation will be possible only if the initial period (corresponding to the acyclic pattern chain) is considered and extended before the other possible periods. For instance in Figure 8, pattern q should be considered as a non-closed suffix of pattern d, and pattern c should therefore be discovered before pattern q. This shows that *the sequence needs to be scanned in a chronological way* and justifies therefore the incremental approach followed by the algorithmic realisation of the modelling, which will be presented in more detail in Section 4.

3.3 The figure/ground rule

Another kind of redundancy appears when occurrences of a pattern (such as the melodico-rhythmic pattern c in Figure 10, representing a decreasing third with 8th note value) are superposed to a cyclic pattern (b'), such that the pattern (c) is more specific than the cycle period

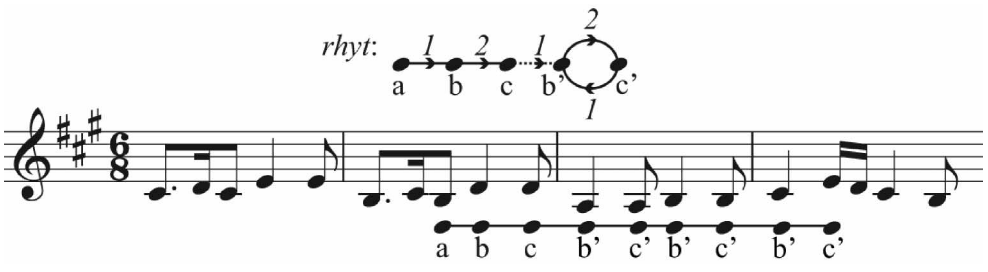

Fig. 9. The listening of successive repetitions of pattern c leads to the induction of its cyclicity, hence to an oscillation between states b' and c'.

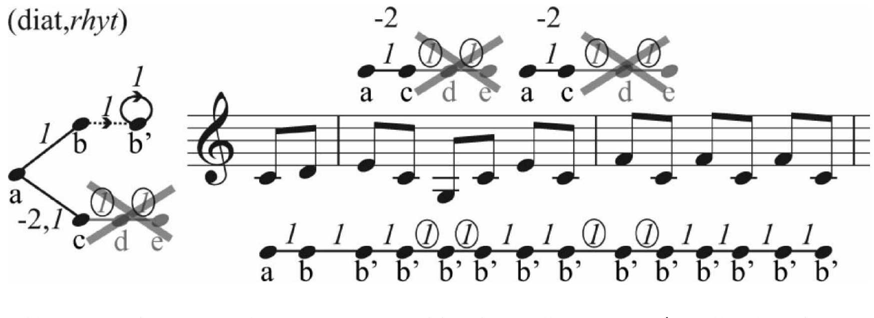

Fig. 10. Pattern c is a specific *figure* above a *background* generated by the cyclic pattern b'. Following the *Gestalt* rule of figure against ground, the figure cannot be extended (into d) by a description that is identical to the background.

(b', which simply represents a repetition of 8th notes). In this case, the intervals that follow these occurrences are identical, since they are related to the same state (b') of the cyclic pattern. Logically the pattern could be extended by the successive extensions of the cyclic patterns (leading to patterns *d*, *e*, etc.). This phenomenon, which may frequently appear in a musical piece, would lead to another combinatorial proliferation of redundant structures if not correctly controlled by relevant mechanisms.

On the contrary, following the *Gestalt* Figure/Ground rule, listeners tend to perceive the pattern *c* as a specific *figure* that emerges above the periodic *background*. Following this rule, the figure cannot be extended (into *d*) by a description that can be simply identified with a background extension.

These redundancy-filtering mechanisms ensure an optimal pattern description. Information is compressed without any loss, since all the discarded structures can be implicitly reconstructed. The filtering of redundant structures ensures clear results and in the same time decreases the combinatorial complexity of the process.

3.4 Musical transformations

One major limitation of the first version of the modelling, as presented in previous sections, is that only repetitions of sequences of notes that are immediately successive could be detected. Yet repeated patterns are often ornamented: the addition of secondary notes can in particular emphasise the important notes of patterns. Figure 11 displays, for instance, a melodic phrase, and one possible ornamentation of it. To some of the notes of the original phrase are added secondary notes (displayed with smaller size in the score) that are located in the neighbourhood of the primary notes, both in time and pitch dimensions.

Solutions to this problem, based on optimal alignments between approximate repetitions using dynamic programming and edit distances, have been proposed (Rolland, 1999; Dannenberg & Hu, 2002). New algorithms are being integrated in our modelling in order to automatically discover, from the surface level of musical sequences, musical transformations revealing the sequence of pivotal notes forming the deep structure of these sequences. These mechanisms induce new connections between non-successive notes, transforming the *syntagmatic chain* of the original musical sequence into a complex *syntagmatic graph*. The direct application of the pattern discovery algorithm on this syntagmatic graph enables the detection of ornamented repetitions. The integration of these new mechanisms is not completely achieved and will not be detailed in this article.

4. Detailed description of the algorithm

4.1 Incremental approach

Patterns and their occurrences are discovered in an incremental way, but in the same time through a chronological scanning of the successive notes of the musical sequence.

4.1.1 Associative memory

The basic principles of our algorithm, focused on an exhaustive discovery of repeated patterns, rely on *associative memory*, i.e. the capacity of associating items that feature similar properties. The associative memory is modelled using tables related to the different musical parameters (i.e. melodic and rhythmic dimensions). A first set of tables store the intervals of the piece with respect to their values along each different musical dimension. For instance, two tables (Figure 12, at the right of node pattern *a*) store the intervals of the score according to their diatonic and rhythmic values. The melodic table shows that the first interval of each bar share a same diatonic value $diat = +1$, and the rhythmic table indicates another identity $rhyt = 1.5$.

The associative memory is formalised in the pseudo-code listed in the annex of the article as a set of hash-tables `AssociativeMemory(Pattern)` related to each `Pattern`, which associative with each value of each parameter – denoted `(Parameter:Value)` – the list of contexts sharing this value. These contexts will be represented by a couple `(Occurrence, Note)`, i.e. the `Occurrence` of the `Pattern` defining that particular context, and the `Note` that follows the `Occurrence`, such that the interval between the `Occurrence` and the `Note` contains the `Parameter Value.`

4.1.2 Pattern discovery

Intervals sharing a same value form occurrences of an elementary pattern that simply represents this particular interval parameter. The elementary pattern is represented

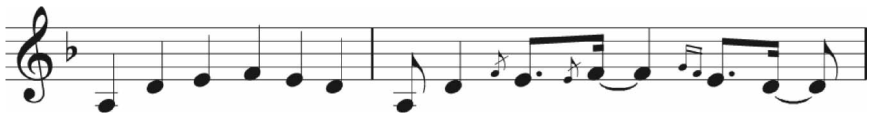

Fig. 11. A melodic phrase and an ornamented version of it.

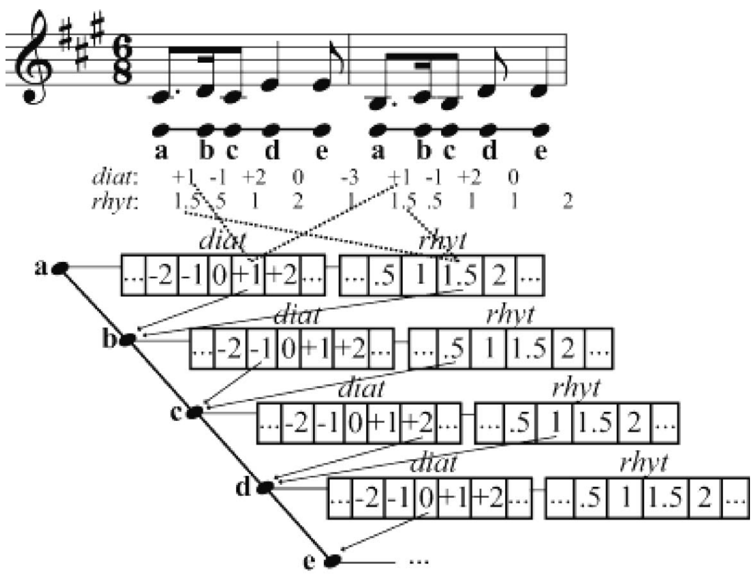

Fig. 12. Progressive construction of pattern *abcde* and of its two occurrences, by storing the intervals in tables associated with each successive state of the pattern.

as a child (*b* in Figure 12) of the root of the pattern tree (*a*). The description of the pattern is the list of parametric identities (arrows in the figure from the previously considered table indexes) and the pattern class is the set of pattern occurrences. Each time a new pattern is created, new tables (at the right of node b) store all the possible intervals that immediately follow the occurrences of the new pattern (b). When any identity is detected in these new tables, a new pattern is created as an extension of the previous one (c, as an extension of b), and is represented as a child in the pattern tree, and so on.

This algorithm enables a progressive discovery of the successive extensions of each pattern, either homogeneous or heterogeneous (as defined in Section 2.2): the selection of musical dimensions defining each successive extension of a pattern may vary. For instance, in Figure 12, the last extension of pattern *e* is simply melodic since the rhythm of the last interval in each occurrence is different. Besides, additional constraints have been integrated in order to ensure a minimal continuity along these variable successive musical dimensions.

The incremental approach proposed here enables a multi-dimensional adaptive discovery of patterns. The use of hash-tables ensures the computational efficiency of the pattern discovery process: thanks to the associative memory, remembering of old similar contexts does not need a search through the score.

4.2 Chronological pattern construction

As justified in section 3.2.3, the pattern discovery process needs to be chronological: the main routine of the algorithm – called `ChronologicalPass` and described in the appendix – consists in a single pass through the musical sequence, from the first note n_i to the last note n_N. Each successive note launches a set of operations – listed in the function `AnalyseNewNote` described in the appendix – insuring the integration of the new note to the structures previously discovered and the creation of new structures. The detailed chain of operations contained in `AnalyseNewNote` is presented below.

4.2.1 Note pattern instantiation

As explained previously, each new note n_i is an occurrence of the note-pattern (represented by state *a* in Figure 13). In this way, each note initiates its own pattern occurrence tree, which will contains all the possible occurrences starting from this note.

4.2.2 Top-down traversal of the specificity graph

The interval between the two last notes n_{i-1} and n_i is constructed and is considered as a candidate extension of all the possible pattern occurrence chains concluded by the previous note n_{i-1}. We have seen that, following the closure principle, the selection of each pattern candidate depends on the support of their most specific patterns. Therefore the set of pattern occurrences concluded by the previous note n_{i-1} – which forms a graph called *Specificity Graph* – needs to be traversed in a decreasing order of specificity.

For instance, in Figure 13, at step $i = 9$, patterns *d* and *c* can both be extended by notes n_9 since the occurrences of both patterns are followed by the same

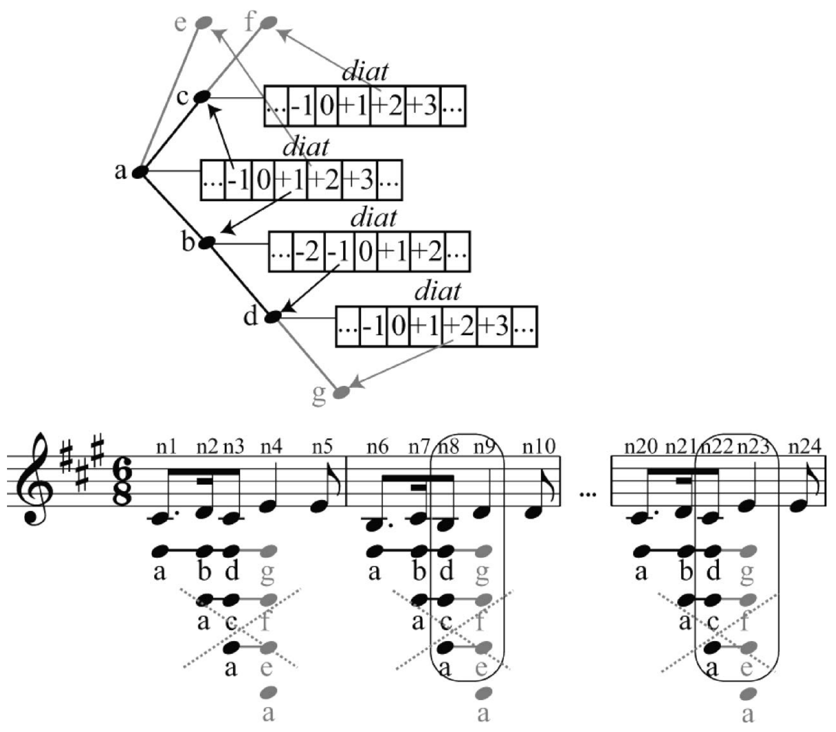

Fig. 13. Chronological analysis of the piece. When considering note n9, patterns *e*, *f* and *g* are discovered. When considering note n23, these patterns are simply recognized.

description (*diat* = + 2). However, the resulting pattern *f* is non-closed since it has same support (2) than its super-pattern *g*, and should therefore not be created. That is why these patterns need to be considered in a decreasing order of specificity: first extending the most specific pattern *d* and then extending, if possible, the less specific pattern *c*.

It should be remarked however that, following our chronological approach, a new pattern class is inferred during the discovering of one of its occurrence. In this context, some of the more specific patterns do not apply to this occurrence at all, and should not be considered here. For instance, for a specific occurrence of a rhythmic pattern, the set of all possible melodico-rhythmic patterns of same rhythm should not be considered entirely. More precisely, the assessment of the redundancy of a candidate pattern simply depends on the pattern *occurrences* more specific than the occurrence currently considered and concluded by the same note.

For each successive pattern occurrence P_{i-1} (and in parallel each corresponding pattern *P* of which P_{i-1} is an occurrence) found during this traversal (at step $i = 9$ in Figure 13, successively occurrences of *d*, *c* and *a*), two tests are operated, namely pattern *discovery* and pattern *recognition*, corresponding to the two subroutines `Discover` and `Recognise` described in the appendix. But first of all, the interval $n_{i-1}\ n_i$ ($n_8\ n_9$ in Figure 13) is memorised in the associative memories related to the candidate patterns (*d*, *c* and *a*), as described in the function `Memorise` in the appendix.

4.2.3 Pattern discovery

As explained in Section 4.1.2, any identity along one or several parameters implies the possible extension of pattern *P* with a new child *D* associated with this identified description. In Figure 13, for instance, at note n_9, pattern *c* can be extended into *f* since the previous occurrence at note n_3 accepts the same extension along the description $diat = +2$. The closure condition should however apply: there should not exist a more specific pattern *R* of same support. In Figure 13, the extension *f* is in fact aborted since the more specific pattern *g* has same support (2). The set of more specific patterns *R* is constructed by extending the corresponding set of patterns *Q* more specific than the original pattern *P* by the same element n_i. Hence pattern *g* is retrieved as an extension of pattern *d*, which is more specific than *c*.

A pattern that is considered as redundant at one moment of the musical sequence may become non-redundant once it appears alone, without the more specific description, at a later stage of the analysis. For instance, in Figure 7, when only the two first occurrences are analysed, both patterns having same support, only the more specific pattern *e* should be explicitly represented. But the less specific pattern *i* will be represented once the last occurrence is discovered, as it is not an occurrence of the more specific pattern *e*.

It should be remarked that, on the other hand, even when a pattern *x* is considered as a non-redundant suffix of another pattern y, its extension x' may, on the contrary, become redundant. This happens when the pattern class of x' is strictly included into the pattern class of x and becomes equal to the pattern class of a more specific pattern y'. For this reason, the redundancy of a pattern should be checked at every phase of its extension.

Each update of the pattern tree by the extension of a pattern P by a new child D (such as the extension of d into g in Figure 13) is associated with an update of the pattern occurrence trees: the current pattern occurrence chain P_{i-1} (occurrence of d concluded by note n_8) is extended with a new child D_i (child g associated with note n_9), occurrence of D. Moreover, all the other previous pattern occurrence chain P_{j-1} are also extended with a new child D_j, every time the interval n_{j-1} n_j complies with the description of D_i. For instance, in Figure 13, the occurrence of pattern d concluded by note n_3 is extended with a child g associated with note n_4.

When the following note n_{i+1} (n_{10}) will be considered, the new interval n_i n_{i+1} ($n_9 n_{10}$) will be automatically memorised in the associative memory of the new patterns D (g) induced by the previous note n_i. The interval n_j n_{j+1} (n_4 n_5), on the contrary, could not be stored in the same way, since, at the moment this interval was considered, the pattern D was not already discovered. Therefore, each time a new pattern is discovered, the interval following each older occurrence[3] should be memorised in the associative memory of this new pattern.[4]

4.2.4 Pattern recognition

Each child C of pattern P is successively considered. When the new interval n_{i-1} n_i complies with the description of the child C, the pattern occurrence P_{i-1} is extended with a new child C_i, linking pattern C to the new note n_i. For instance, in Figure 13 for note n_{23}, the occurrence of pattern d concluded by the previous note n_{22} can be extended into an occurrence of its child g as the description of the new interval n_{22} n_{23} complies with the child description ($diat = +2$).

In this subsection the pattern discovery process has been described before pattern recognition since a pattern should first be created before being recognised. However, during the incremental process of the analysis, as presented in the function `AnalyseNewNote` in the appendix, the order is reversed: the pattern recognition test, quite straightforward, can be operated before the pattern discovery test.

4.3 Compact score representation

The pattern description has been reduced even further through a selection of maximally specific pattern occurrences: When a pattern occurrence is discovered (pattern e in Figure 7), all the occurrences of less specific patterns (pattern i) may be ignored, since they do not bring additional information, and can be directly deduced from the most specific pattern occurrence (e) and from the specificity relation (between e and i).

The less specific description should be taken into account implicitly though, because their extensions may produce specific descriptions. For instance in Figure 14, groups 1 and 3 are occurrences of pattern h, and groups 3 and 4 are occurrences of pattern d. Since pattern d is more specific, the less specific pattern h does not need to be associated with group 4. However in order to detect groups 2 and 5 as occurrences of pattern l, it is necessary to implicitly consider group 4 as an occurrence of pattern h. Hence, even if pattern h, less specific than d, was not explicitly associated with group 4, it had to be implicitly reconstructed in order to construct pattern l. Implicit information is retrieved through a traversal of the pattern network along specificity relations.

4.4 Generalisation of patterns

New patterns can be discovered through a generalisation of already known patterns. In bar 7 of Figure 15, the two first notes form an occurrence of pattern h. The third note cannot however be associated with the known extension of pattern h into pattern i, because the melodic description $diat = 0$ does not fit the context. However, as the rhythmic description $rhyt = 2$ fits, a new extension j is discovered through a generalisation of pattern i by removing the non-matching description.

The less specific patterns, although not explicitly represented in the analysis, should be updated if necessary. In particular, when a pattern is generalised, the more general patterns should be generalised too. For instance, as i has been generalised into j, c should also be generalised into k in the same way. Hence, at next bar (8) the general pattern k is simply recognised.

[3]At first sight, since a pattern is discovered after a repetition of two of its occurrences, this problem concerns only one occurrence. However, some patterns, such as pattern i in Figure 7, may be discovered only when repeated more than twice. In this case, the problem concerns all the occurrences before the last discovered occurrence. The continuation of the last occurrence, on the contrary, will be memorized following the standard procedure, i.e., once the new interval will be considered.

[4]The cognitive relevance of such mechanisms may be questioned, since the relation between the older occurrences and their next intervals may be forgotten if these occurrences do not belong to the short term memory any more. In a more realistic modeling, the memorizing of these old continuations may be carried out through multiple successive scanning of the whole score. Indeed, a listener usually needs to listen musical pieces several times before actually catching the whole structures. Nevertheless, these results can be obtained more rapidly using the less realistic mechanism.

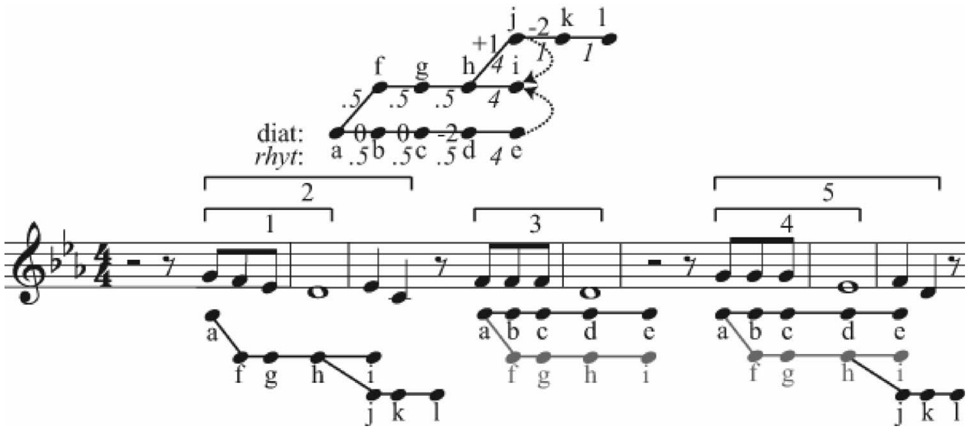

Fig. 14. Group 4 can be simply considered as occurrence of pattern *d*. However, in order to detect group 5 as occurrence of pattern *l*, it is necessary to implicitly infer group 4 as occurrence of pattern *h* too.

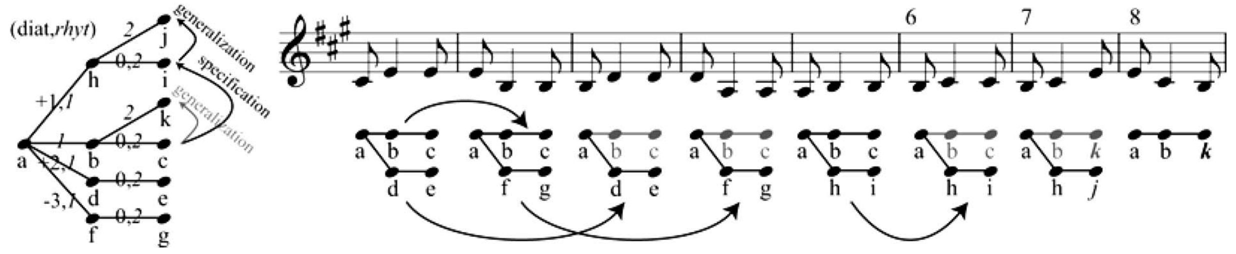

Fig. 15. Progressive discovery of the pattern repetitions on the score and the resulting pattern tree (at the left of the score). Pattern descriptions in gray are less specific than simultaneous descriptions in black, and should not therefore be represented explicitly. However, the generalization of pattern *i* (bar 6) into pattern *j* (bar 7) leads to the implicit generalization of pattern *c* into pattern *k*, than can therefore be immediately identified in bar 8.

4.5 General and specific cycles

The specificity relation defined in Section 3.1 had to be applied to cyclic patterns too: a cyclic pattern *C* would be considered as more specific than another cyclic pattern *D* when the sequence of description of pattern *D* is included in the sequence of descriptions of pattern *C*. Here too, this concept of specificity plays a pivotal role in music perception and enables a sound algorithmic processing of music. In Figure 16, the seven first notes of the cycle oscillate around the cyclic PC b'-c' simply composed of an oscillation between two rhythmic values of 8th note and quarter note (*rhyt* = *1* and *2*), the second interval also associated with a unison interval (*diat* = 0). Then a more specific cycle d$'$-e' includes an ascending interval (*diat* = + 1), and is generalised after four notes into cycle d$''$-f' that does not feature the unison interval any more. Moreover, following the rule of generalisation of generalised patterns explained in previous section, the more general cycle b'-c' needs to be generalised into a cycle b''-g' where the unison interval has been discarded. The integration of this phenomenon into the model helps insuring the relevance of the results and avoiding numerous unwanted combinatorial redundancies.

5. Results sensibly close to perceived structures

5.1 Implementation

This model, called *kanthus*, was first developed in *Common Lisp* as a library of *OpenMusic* (Assayag et al., 1999). A new version in *C Language* will be included in the next version 2.0 of *MIDItoolbox* (Eerola & Toiviainen, 2004). The model can analyse monodic musical pieces (i.e. pieces composed of a series of non-superposed notes) and highlight the discovered patterns on a score. Rhythmic values are obtained through simple quantification operations and scale degree parameters are computed through a straightforward mapping between pitches values and scale degrees.

5.2 Some results

The model has been tested using different musical sequences taken from several musical genres (classical music, pop, jazz, etc.) and featuring various level of complexity, from very simple melody like *Au clair de la lune* (French folk song) or the first bars of Beethoven's *Fifth Symphony*, to more complex pieces. The experiment has been run using version 0.7 of *kanthus* on a 1-GHz PowerMac G4. This section presents the analysis of three pieces of various styles: a medieval *Geisslerlied*, a Bach *Invention*, and a Tunisian modal improvisation. The contour dimension, although integrated in the modelling and taken into consideration in simple musical examples, is discarded for these more complex pieces. Indeed, the integration of contour in a general framework requires the handling of particular problems, such as the limitation of contour identification to short term memory (Dowling & Harwood, 1986).

5.2.1 Geisslerlied

Figure 17 presents the resulting analysis of a medieval song called *Geisslerlied* that the linguist Nicolas Ruwet (1987), in one of the first and most famous attempts to model motivic analysis, proposed as a first application of his method. Our model is the first computational system able to offer a relevant and compact analysis of this piece. The piece considered here is however a slight simplification of the actual piece presented in (Ruwet, 1987), which includes several local motivic variations that could not

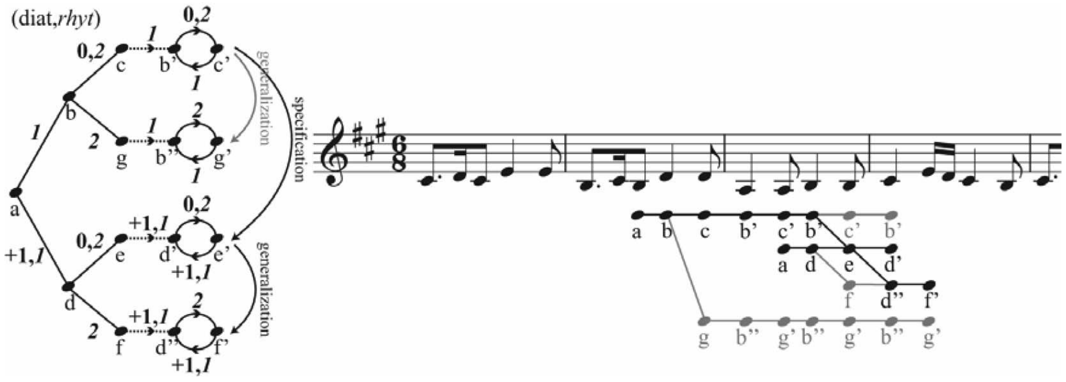

Fig. 16. Actually, complex cyclic patterns are perceived: a first oscillation between a 8th note and a quarter note associated with a unison interval (b'-c') is specified through the integration of an ascending interval associated with the 8th note (d'-e'). The cycle is then generalized due to the absence of the unison (d''-f') and cycle a'-b' is also generalized, for the same reason, into b''-g'.

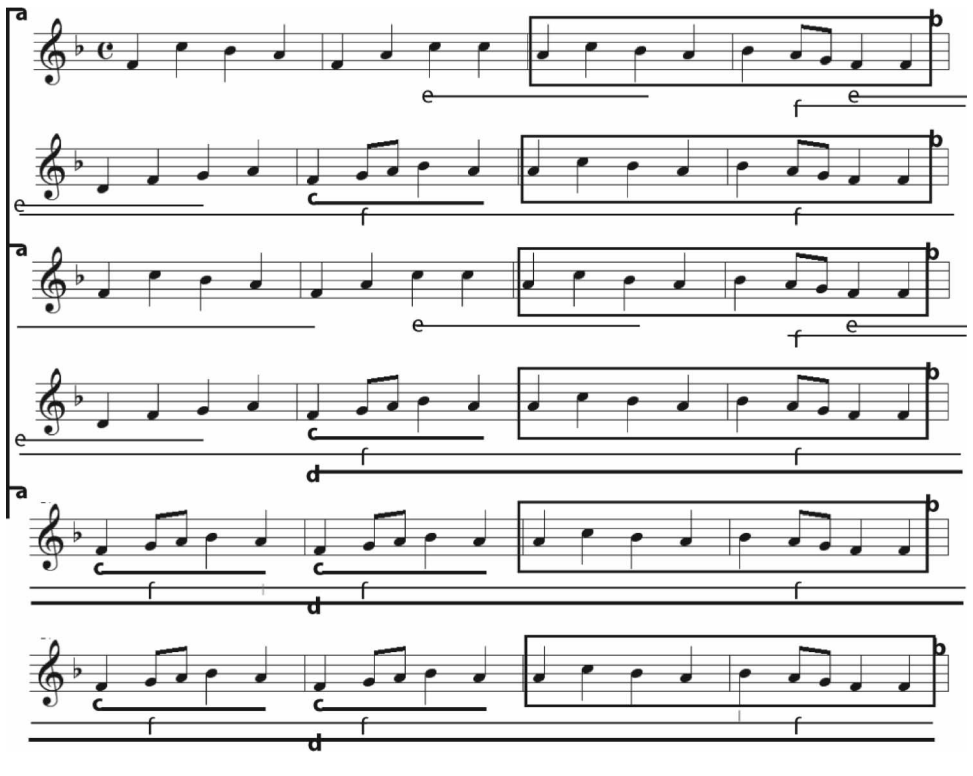

Fig. 17. Analysis of a *Geisslerlied* (slightly simplified).

be detected for the moment. These complex musical transformations will be further considered once the mechanisms presented in Section 3.4 will be implemented.

Pattern *a* corresponds to the exact repetition of the first eight bars, implying the expectation of a third occurrence (indicated by the third *a* graduation) finally aborted. Pattern *b* represents the identical 2-bar long ending of each line of the score. The second part of the piece, after the two occurrences of patterns *a*, features two successive repetitions of pattern *d*, which consists of two repetitions of pattern *c* concluded by the *b* ending. The ending of the second occurrence of *a* contains a suffix of *d* – a succession of patterns *c* and *b* – which is therefore automatically aggregated to the two successive occurrences of *d* forming altogether a cyclic pattern. As indicated in Section 3.2.2, the indicated graduations of the cyclic pattern are not meant to correspond to actual segmentation detected by listeners. A relevant segmentation of the cyclic pattern (most probably at the beginning of each line) would require the incorporation of new mechanisms, based for instance on the expected segmentation induced by patterns *a* and *b*. Finally a short pattern *e* is also detected, as well as a cyclic pattern *f* indicating the simple alternation between 8th notes and quarter notes. The resulting pattern tree contains 158 nodes, 24 different cyclic patterns have been discovered, and the pattern occurrence trees contain 1367 nodes in all. The numerous cyclic patterns result from particular structural configurations – such as the successive repetitions of pattern occurrences, the progressive transformations of cyclic patterns, and the successive repetitions of same simple rhythms – but may easily be filtered if necessary. The analysis took 28 seconds of CPU time.

5.2.2 Bach Invention in D minor

The algorithm has been applied to a melodic analysis of a complete two-voice *Invention* by J.S. Bach. Figure 18 shows the analysis of the 21 first bars. The repetition of ascending quarter notes in bars 3 and 4 has not been detected, as the contour dimension was not considered in the experiment. The proposed computational modelling offers high precision and recall that has never been reached by previous approaches.

The analysis of the whole piece, which contains 283 notes, took 171 seconds, and generated a pattern tree with 315 nodes and 11 cyclic patterns, and the occurrence trees contain 6409 nodes in all. In fact, the rhythmic configuration causes a proliferation of irrelevant structures. In order to avoid these redundant structures, a concept of meta-pattern of patterns needs to be integrated into the framework. If rhythmical dimensions are not taken into account, the analysis takes only 12 seconds, the pattern tree contains 102 nodes and 11 cycles, and the pattern occurrence trees contains 1064 nodes.

5.2.3 Analysis of Arabic modal improvisation

The model has also been applied to the analysis of Arabic modal improvisation, where melodies are highly ornamented by the addition of secondary notes. Therefore, as explained in Section 3.4, the detection of repeated patterns requires the integration of mechanisms that transform the original syntagmatic chains into graphs in order to retrieve the core melodic structures. A first experimental test of these new mechanisms have been attempted with the analysis of Mohamed Saada's improvisation of *Istikhbar Mhayyer Sika*, displayed in Figure 19. The discovered structures are represented below the score. Patterns are designated by a sign (1, 2, 3, 4, 5, + and −) on the left of the corresponding lines. The notes associated with each pattern occurrence are represented by squares vertically aligned to the notes. These squares represent therefore the successive states along the pattern occurrence chain, similarly to the representation displayed in Figure 4.

Pattern '−' represents a simple sequence of notes of continuously decreasing pitch heights, and pattern '+' represents a sequence of notes of continuously increasing pitch heights. Patterns 1 to 5 are sequences repeated several times in the improvisation. Black squares represent the starting of new occurrences, and white squares the successive states along the pattern chain. Grey squares

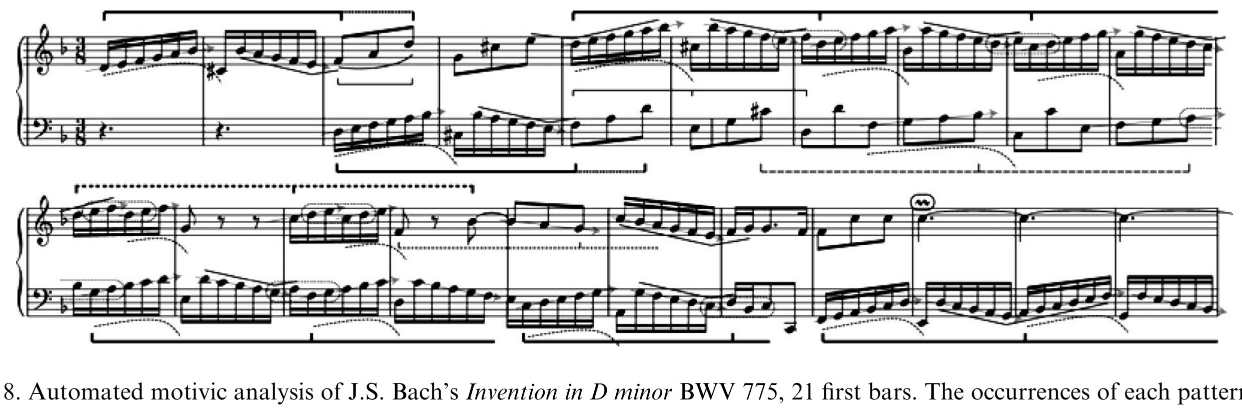

Fig. 18. Automated motivic analysis of J.S. Bach's *Invention in D minor* BWV 775, 21 first bars. The occurrences of each pattern class are designated in a distinct way.

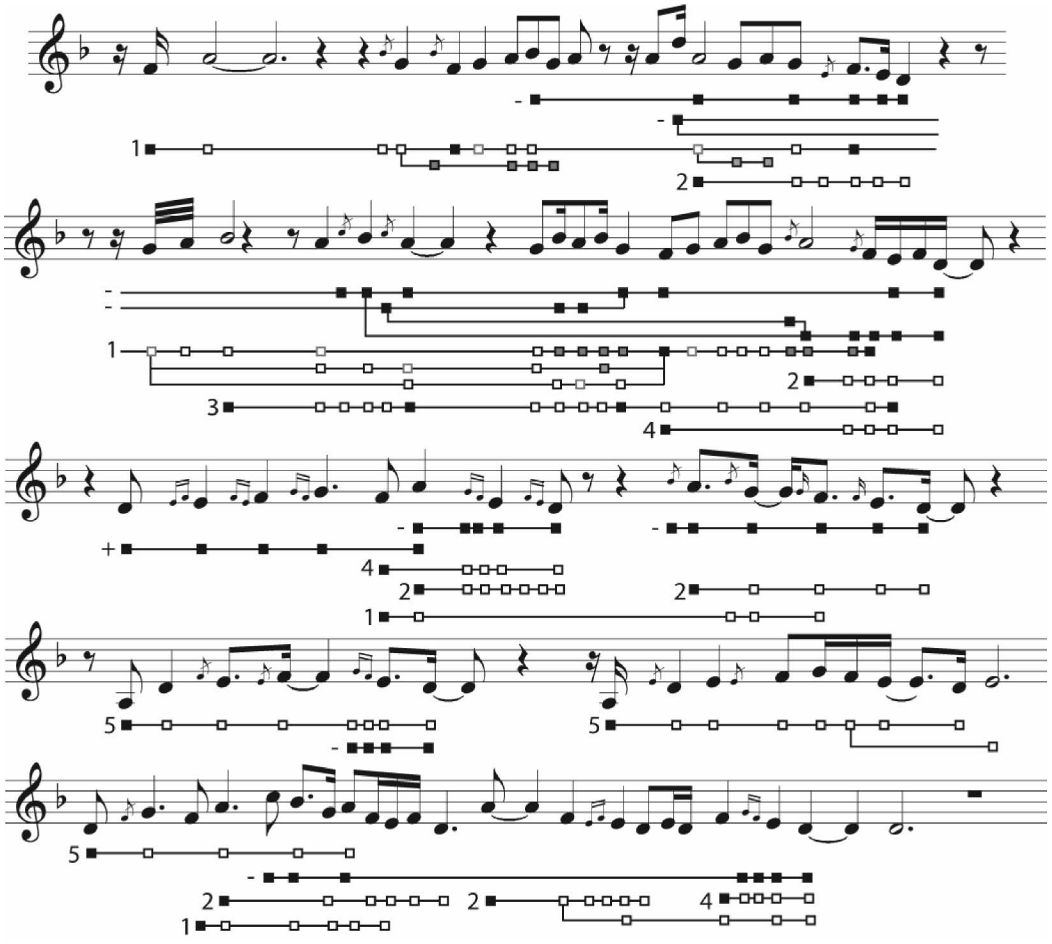

Fig. 19. Analysis of the beginning of *Istikhbar Mhayyer Sika* improvised at the Nay flute by Mohamed Saada (transcribed by Mondher Ayari).

correspond to optional states that do not exist in all the occurrences of the pattern. Finally, multiple branches designate multiple possible paths for one same pattern occurrence. The improvisation is built on the specific mode *Tba' Mhayyer Sika D*, characterised by the use of a specific set of notes (D, E, F, G, A, Bb, C) and a specific melodic figure, which corresponds exactly to pattern 2. The beginning of the improvisation is also based on the successive repetition of pattern 1, which corresponds to a periodic melodic curve starting from note F and ending to the same note F, which is therefore a pivotal note of the improvisation. The second line of the improvisation is characterised by the successive repetition of pattern 3, which is a little melodic line progressively transposed. Pattern 4 corresponds to another important melodic profile associated with pattern 2. Finally the two last lines of the improvisation are characterised by the repetition of pattern 5.

Besides these discovered structures, the application of the pattern discovery algorithm in the general syntagmatic graph leads to combinatorial explosion of redundant patterns not fully controlled yet, which will need further researches.

5.3 Algorithm complexity

The algorithm complexity may be considered in two respects. First, with respect to the complexity of discovered structures: proliferation of redundant patterns, for instance, would lead to combinatorial explosion, since each new structure needs proper processes evaluating the interrelationships with the other structures, and inferring their possible extensions. Hence, a maximally compact description ensures both clarity and relevance of the results and limitation of combinatorial explosion.

Complexity may be considered also with respect to the technical implementation of the modelling. The proposed algorithms are for the time being only partially optimised. The mechanisms of redundancy filtering, as presented in the article, require an important number of checks. Due to the complexity of the interrelations between the multiple mechanisms, the computational expense of these checks and the memory consumption have not been evaluated in detail yet.

6. Discussion and future work

This study has shown that the musical patterns actually discovered by the listeners cannot be reduced to simple mathematical objects. The actual complex strategies undertaken during the listening process need to be modelled as carefully as possible. The model proposed in this paper is a first attempts towards this objective. The different mechanisms proposed here result from a progressive building of the system. At each stage of this building, the bad behaviours of the analysis process were explained through the induction of causes that needed to be as simple and general as possible. Following these explanations, the model was improved through a modification of the mechanisms and the addition of new ones. These levels of precision or of perceptive relevance have never been reached by previous approaches, which include a numerous set of redundant patterns such as suffixes or redundant extensions. This shows the necessity of adaptive redundancy filtering as proposed in this article. However, the analyses remain significantly restricted, as numerous aspects of musical expression have not been taken into account yet.

6.1 Future work

6.1.1 Integrating Gestalt segmentation mechanisms

The structures currently found are based solely on pattern repetitions. Should be added, as considered in Section 1.2, the alternate mechanism based on merging of notes closed in time or pitch domain, and, reversely, on segmentation between distant notes, following *Gestalt* rules of proximity and similarity (Lerdahl & Jackendoff, 1983; Temperley, 1988; Cambouropoulos & Tsougras, 2004). Although this rule plays a significant role in the perception of large-scale musical structures, there is no common agreement on its application to detailed structure, because it highly depends on the subjective choice of musical parameters used for the segmentations (Deliège, 1987). In our approach, we propose to investigate the interdependencies between the two rules of pattern discovery and *Gestalt* segmentation. For instance, a pattern repetition may be masked due to an important temporal gap within one of the occurrences.

6.1.2 Polyphonic pattern discovery

Our approach is currently limited to the detection of repeated *monodic* patterns (i.e. sequences of successive notes) in monodic musical pieces. Music, in general, is *polyphonic*: it can contain simultaneous notes forming chords, in particular, and simultaneous monodic lines forming different voices. Researches have been carried out in this domain (Conklin & Anagnostopoulou, 2001; Dovey, 2001; Meredith et al., 2002; Meudic & Saint-James, 2004), mainly focused on the discovery of repeated exact patterns along different pre-specified dimensions. In our approach, we are currently developing rules of automated discovery of melodic lines inside polyphonic sets of notes (or *stream segregation*), based on cognitive heuristics. Our study will then focus on the interactions between pattern discovery and stream segregation. We will then extend the scope by considering pattern of chords, which will need the definition of a more general concept of interval between successive chords. Could also be considered patterns composed of successions of groups of notes (Deutsch & Feroe, 1981; Conklin & Anagnostopoulou, 2006; Lartillot & Saint-James, 2004).

6.2 Applications

6.2.1 Cognitive modelling

This study showed that musical patterns result from numerous interdependent mechanisms that need to be carefully modelled within a conceptual network. The different processes are implemented as basic operators applied to each successive phase of the construction of the structure. The stability of the whole system depends on the good definition of each elementary operator: a little hidden default may lead to general chaotic behaviour and combinatorial explosion. Because of the difficulty to control the whole mechanisms and to ensure the relevance of discovered patterns, we assume that a modelling able to offer satisfying results may present some analogy with the actual cognitive processes ruling human listening process. The resulting model will hence be offered to cognitive validations and improvements with the help of experimental psychology.

6.2.2 Industrial applications

The automated discovery of repeated patterns may lead to interesting applications. A new kind of similarity distance between musical pieces may be defined, based on these pattern descriptions. A music database could then be browsed using this pattern-based similarity distance: from a given musical piece, a user may find pieces containing similar patterns. The automated pattern description of musical database may also enable an improvement of pattern matching algorithm: when a user looks for a specific pattern in a music database, the search, that should initially be undertaken throughout the whole database, can be reduced to the set of characteristic patterns that have been discovered during the initial analysis.

Acknowledgements

These researches are carried out at the University of Jyväskylä with Petri Toiviainen, in close collaboration between computer science, cognitive psychology, musicology and ethnomusicology, in particular within the context of a collaborative project with Stephen McAdams (CIRMMT, Montreal), Mondher Ayari and Gérard Assayag (IRCAM) funded by the French Academy (CNRS). The paper has been improved thanks to valuable recommendations offered by the reviewers.

References

Assayag, G., Rueda, C., Laurson, M., Agon, C., & Delerue, O. (1999). Computer assisted composition at Ircam: From Patchwork to OpenMusic. *Computer Music Journal*, *23*(3), 59–72.

Cambouropoulos, E. (1998). Towards a general computational theory of musical structure. Doctoral dissertation, University of Edinburgh, 1998.

Cambouropoulos, E., Crochemore, M., Iliopoulos, C., Mouchard, L., & Pinzon, Y. (2002). Algorithms for computing approximate repetitions in musical sequences. *Journal of Computer Mathematics*, *79*(11), 1135–1148.

Cambouropoulos, E. & Tsougras, C. (2004). Influence of musical similarity on melodic segmentation: Representations and algorithms. *Proceedings of the International Conference of Sound and Music Computing*. Paris: Ircam, pp. 147–152.

Conklin, D. & Anagnostopoulou, C. (2001). Representation and discovery of multiple viewpoint patterns. *Proceedings of the International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 479–485.

Conklin, D. & Anagnostopoulou, C. (2006). Segmental pattern discovery in music. *INFORMS Journal of Computing*.

Cook, N. (1987). *A guide to musical analysis*. London: Dent.

Cope, D. (1991). *Computer and musical style*. Oxford: Oxford University Press.

Chrochemore, M. (1981). An optimal algorithm for computing the repetitions in a word. *Information Processing Letters*, 12(3), 244–250.

Dannenberg, R. & Hu, N. (2002). Pattern discovery techniques for music audio. In: M. Fingerhut (Ed.), *Proceedings of the International Conference on Music Information Retrieval*. Paris: Ircam.

Deliège, I. (1987). Grouping conditions in listening to music: An approach to Lerdahl and Jackendoff's grouping preference rules. *Music Perception*, *4*(4), 325–350.

Deutsch, D. & Feroe, J. (1981). The internal representation of pitch sequences in tonal music. *Psychological Review*, *88*(6), 503–522.

Dovey, M.J. (2001). A technique for 'regular expression' style searching in polyphonic music. In: J.S. Downie & D. Bainbridge (Eds.), *Proceedings of the International Conference on Music Information Retrieval*. Bloomington: Indiana University.

Dowling, W.J. & Harwood, D.L. (1986). *Music cognition*. London: Academic Press.

Eerola, T. & Toiviainen, P. (2004). MIR In Matlab: The MIDI Toolbox. *Proceedings of the International Conference on Music Information Retrieval*. Barcelona: Universitat Pompeu Fabra.

Ganter, B. & Wille, R. (1999). *Formal concept analysis: Mathematical foundations*. Berlin: Springer-Verlag.

Lartillot, O. (2004a). A musical pattern discovery system founded on a modelling of listening strategies. *Computer Music Journal*, *28*(3), 53–67.

Lartillot, O. (2004b). *Fondements d'un système d'analyse musicale suivant une modélisation cognitiviste de l'écoute*. Doctoral dissertation, University of Paris 6, 2004.

Lartillot, O. & Saint-James, E. (2004). Automating motivic analysis through the application of perceptual rules. *Music query: Methods, strategies, and user studies (Computing in Musicology 13)*. Cambridge, MA: MIT Press.

Lerdahl, F. & Jackendoff, R. (1983). *A generative theory of tonal music*. Cambridge, MA: MIT Press.

Meredith, D., Lemström, K., & Wiggins, G. (2002). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, *31*(4), 321–345.

Meudic, B. & Saint-James, E. (2004). Automatic extraction of approximate repetitions in polyphonic MIDI files based on perceptive criteria. In: U.K. Will (Ed.), *Computer music modelling and retrieval*. Berlin: Springer-Verlag, pp. 124–142.

Mongeau, M. & Sankoff, D. (1990). Comparison of musical sequences. *Computers and the Humanities*, *24*, 161–175.

Nattiez, J.-J. (1990). *Music and discourse: Towards a semiology of music*. Princeton: Princeton University Press.

Reti, R. (1951). *The thematic process in music*. New York: Macmillan.

Rolland, P.-Y. (1999). Discovering patterns in musical sequences, *Journal of New Music Research*, *28*(4), 334–350.

Ruwet, N. (1987). Methods of analysis in musicology. *Music Analysis*, *6*(1–2), 4–39.

Temperley, D. (1988). *The cognition of basic musical structures*. Cambridge, MA: MIT Press.

Zaki, M. (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 462–478.

Appendix: Pseudo-Code of the basic mechanisms[5] of the modelling

ChronologicalPass:

```
FOR each successive Note of the musical sequence
  CALL AnalyseNewNote(Note)
ENDFOR
```

[5]In order to keep the pseudo-code as simple as possible, the mechanisms introduced in Sections 3.3, 4.4, 4.5 and 4.6 are not presented here.

```
AnalyseNewNote (Note):
CREATE an occurrence of NotePattern related to Note
FOR each Pattern Occurrence concluded by the previous note, from the most specific to the less specific ones
  CALL Memorise (Pattern, Occurrence, Note)
  CALL Recognise (Pattern, Occurrence, Note)
  CALL Discover (Pattern,Occurrence, Note)
END FOR
```

```
Memorise (Pattern, Occurrence, Note):
FOR each musical Parameter
  LET Value be the Parameter value related to Note
  ADD the new context (Occurrence, Note) to the AssociativeMemory (Pattern) at corresponding (Parameter:Value) address
END FOR
```

```
Recognise (Pattern, Occurrence, Note):
FOR each Extension of Pattern
  IF Note complies with the Extension description
    CREATE a NewOccurrence of Extension by extending Occurrence with Note
    CALL DetectCyclicity (NewOccurrence)
  END IF
END FOR
```

```
Discover (Pattern, Occurrence, Note):
FOR each possible Description of Note, from the most specific to the less specific ones
  IF Description is not included in an existing extension of Pattern
    LET Intersect be the intersection of the lists returned by AssociativeMemory (Pattern) for each (Parameter:Value) contained in Description
    LET Context be the union of all couples (RecalledOccurence, RecalledNote) that confirm the condition FigureGroundRule (RecalledOccurrence, RecalledPattern)
    IF the number of Contexts is higher than the class of each pattern more specific than (Pattern, Description)
      CREATE a NewPattern as an extension of Pattern with a new state related to Description
      FOR each (RecalledOccurrence, RecalledNote) in Contexts
        CREATE a NewOccurrence of NewPattern by extending RecalledOccurrence with RecalledNote
        CALL DetectCyclicity (NewOccurrence)
      END FOR
      END IF
    END IF
END FOR
```

```
DetectCyclicity (NewOccurrence):
LET Pattern be the pattern associated WITH NewOccurrence
IF Pattern is not cyclic
AND IF NewOccurrence is immediately preceded by an occurrence of Pattern or a pattern more specific than Pattern
  CREATE a CyclicPattern related to Pattern
END IF
```

```
FigureGroundRule (Occurrence, Note):
LET PreviousNote be the note that concludes Occurrence
THERE IS NO OtherOccurrence concluded by PreviousNote
  SUCH THAT the pattern of OtherOccurrence be cyclic
  AND Occurrence be more specific than the OtherOccurrence
  AND OtherOccurrence be longer than Occurrence
  AND the description of the extension of the OtherOccurrence be more specific than the description of Note
```