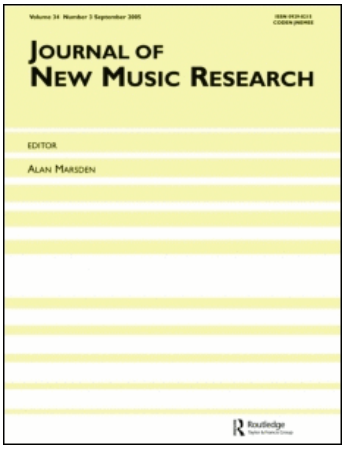
This article was downloaded by: *[Aalborg University]*
On: *11 May 2011*
Access details: *Access Details: [subscription number 912902580]*
Publisher *Routledge*
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



Journal of New Music Research
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713817838

Discovering Patterns in Musical Sequences

Pierre-Yves Rolland

Online publication date: 09 August 2010

To cite this Article Rolland, Pierre-Yves(1999) 'Discovering Patterns in Musical Sequences', Journal of New Music Research, 28: 4, 334 — 350
To link to this Article: DOI: 10.1076/0929-8215(199912)28:04;1-O;FT334
URL: http://dx.doi.org/10.1076/0929-8215(199912)28:04;1-O;FT334

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.informaworld.com/terms-and-conditions-of-access.pdf

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan or sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Journal of New Music Research, 28 (1999), No. 4, pp. 334–350 0929-8215/99/2804-334$15.00
©Swets & Zeitlinger

Discovering Patterns in Musical Sequences

Pierre-Yves Rolland

Laboratoire d'Informatique de Paris
Université Pierre et Marie Curie, France

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

ABSTRACT

Pattern discovery (or 'extraction') in sequences is a very general problem with diverse musical applications ranging from music generating systems to melodic content-based retrieval to music analysis. It naturally fits within the wider problematics of musical (and multimedia) content extraction. In this article, we focus on the automated discovery of patterns in corpuses of melodic sequences. A melodic pattern is defined by a set of either identical or 'equipollent' (i.e., significantly similar) sequence segments. In previous work and articles, we addressed such critical issues in musical pattern discovery as the representation of sequences and of their elements, and the definition of appropriate similarity metrics between (pairs of) sequence segments. We now present a novel pattern extraction algorithm named FlExPat ('FlExible Extraction of Patterns'), which builds upon the concepts and techniques we previously introduced. FlExPat articulates in two phases, passage pair comparison and then categorization. Its theoretical worst-case complexity is quadratic in the corpus' total sequence length, but both running time and required memory are far smaller in practice. FlExPat has been implemented in our Imprology software system. Experimental results, a few of which are detailed here, clearly show FlExPat's qualities and performances.

INTRODUCTION

Musical pattern discovery, and more specifically the *extraction* of melodic or harmonic patterns in given sets of composed or improvised works, is an important problem within the wider context of musical/multimedia content extraction. It has many different applications ranging from music generating systems to multimedia content-based retrieval to music analysis (see Rolland and Ganascia, 1999, for a review and the section on future work for more details about applications to musical information retrieval).

One example of music analysis is the voluminous study by the musicologist T. Owens (1974), in which he characterized the techniques and style of Charlie Parker through the nature and organization of recurrent melodic patterns ('formulae') in the jazz saxophone player's improvised playing. He transcribed about 250 Parker solos and extracted a hierarchically classified lexicon of 193 such patterns from this corpus. Examples of these patterns will be given later on. We have used Owens' corpus and lexicon as validation material for our implemented pattern extraction system. Studies such as Owens' take years to conduct, as the musicologist must, in particular, deal manually with the huge combinatorics inherent to the sequential pattern discovery process. This was a strong motivation for us when seeking to automate the process. Improvised jazz (especially in *BeBop* style) is a particularly interesting kind of musical material for pattern discovery, and also a particularly difficult one, as opposed to, e.g., baroque or classical music. In

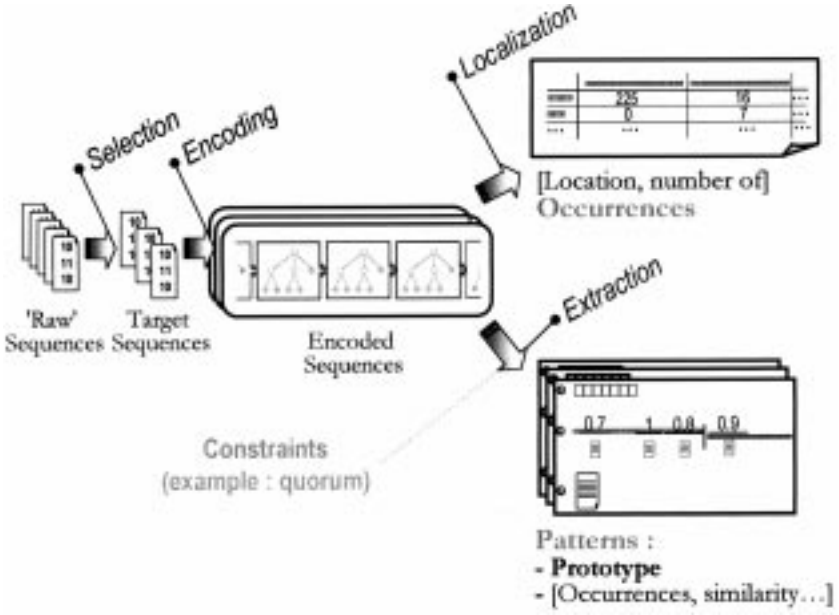


Fig. 1. Schematic of pattern extraction (vs. pattern localization).

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

the computational experiments we have been carrying out, we focused on this kind of material. Many musical examples will be shown in this article which were transcribed from actual jazz improvisations.

FORMALIZATION

The general process of sequential pattern extraction (*sequential data mining*) is schematized in Figure 1. Also depicted is the pattern localization process, which consists of finding occurrences of sequential patterns that are *given* in advance. In machine learning terms, pattern localization, which will not be covered in this article, can be viewed as the supervised counterpart of pattern extraction.

Melodic sequences and patterns

We represent melodies as sequences over generalized alphabets. We extended the notion of an alphabet, which is a finite set of *symbols*, to that of a generalized alphabet — a finite set of entities of any nature, viz. structured representations of musical notes.

Intuitively, a pattern can be viewed as a set of things that display significant resemblance according to some similarity definition. This is an *extensional* view, in contrast to an *intentional* one. In the latter view, one is interested in an entity which represents a class (or category) of significantly similar things. This entity, which is often called a *prototype* of the class, can be defined using various criteria of typicality, cohesion, etc. Based on these classical concepts of categorization/classification (see, e.g., Rosch, 1978), we will now define the key concepts related to sequential pattern discovery.

We define extensionally a *melodic pattern* as a set[1] of melodic passages that share a significant resemblance according to some melodic similarity definition. These passages are called *occurrences* of the pattern, and the 'significant resemblance' relation is dubbed *equipollence*.[2] In particular, given a *corpus*, which is an ordered set of melodic sequences, the occurrences of a melodic pattern are passages from one or several of the melodies. It is important to note that we do not demand that

[1]In the combinatorial string processing field, such a set is called a *block*.
[2]This term, rather than that of *equivalence,* is used — This is because by definition an equivalence relation implies a number of properties, such as transitivity, that appear as excessively stringent in the context in which we are placed.

passages necessarily be *phrases*, in the closure-related sense. In sequential terms, each passage corresponds to a particular contiguous segment of one sequence. Intentionally, a pattern is represented by a *prototype,* which can be an actual melodic passage as is the case for the algorithm presented below; but it can also be a more abstract entity such as a regular expression.

Pattern discovery (or *extraction*) consists of finding all patterns in a given corpus, possibly imposing additional constraints such as a *quorum* threshold fixing the minimum number of different sequences in the corpus in which each pattern must appear (prototype or occurrences). So far, we have focused on monodic (monophonic) material, although a generalization to polyphonies or to harmonic sequences (chords) is entirely possible.

Equipollence is defined by a threshold on numerical similarity values. To test the equipollence between two passages, they are *compared* using a numerical similarity assessment algorithm; if their similarity value is above the threshold, they are determined equipollent.

Music representation

Choosing an appropriate representation for music (melodies, passages, notes, etc.) has long been recognized as critical in any computational approach (Widmer, 1992, 1994), and automated pattern discovery is no exception in that respect (see, e.g., Anagnostopoulou et al., 1999; Cambouropoulos et al., 1999; Rolland and Ganascia, 1996b). In particular, through extensive experimentation (Rolland, 1998), we established empirically that melodic passage comparison requires taking simultaneously into account multiple and cognitively pertinent descriptions of melodies and of their individual notes. The representations used in the usual encodings of melodies, viz. musical score notation and MIDI, are only an extremely restricted version of descriptions. In addition to — or instead of — the initial representation of the melodies' notes (absolute pitch and relative duration, typically), structural information should be taken into account at various abstraction levels, based on the psychology of music (perception, cognition) and on music theory. The relative importances of the possibly redundant simultaneous descriptions differ one from another as a function of the context — a general observation in the study of similarity assessment (Suppes et al., 1990). For instance, a comparison between musical passages, or a pattern extraction from a melodic corpus, may be performed from a more rhythm-oriented viewpoint or, on the contrary, from a more pitch-oriented one.

All of these make music, and especially jazz, a particularly difficult and interesting domain for the computational modeling of 'knowledge-intensive' tasks like sequential pattern discovery.

We have proposed the insertion, within the pattern discovery process shown in Figure 1, of a user-controlled phase where the initial representation is enriched (with *additional descriptions*) — or changed. This enrichment or change is based (generically) on the application-specific domain theory and/or background knowledge. It is on the *final representation* obtained that pattern extraction is carried out (Fig. 2). The redundancy possibly induced by the multiplicity of simultaneous descriptions in the final representation is not problematic, as mentioned earlier. We have automated within our Improlog system (see the section on implementation) this representation enrichment/change phase for melodic sequences, which typically have an initial MIDI-type representation. In the case of music, background knowledge and domain theory are based on psychological works of music perception and cognition, and on music theoretical works. From these, we identified and classified a set of eligible user-selectable musical descriptions at various levels of abstraction, which can be *derived* from the basic (initial) descriptions of melodies and notes/rests.

Additional descriptions are always *derived* from initial descriptions or from any combination of initial and already derived descriptions. For instance, the "BackwardChromaticInterval" description, which is the number of semitones between a note and its predecessor, is computed from the "AbsolutePitch" description of the note under consideration and of its predecessor. The *order* in which the various descriptions are computed during the representation enrichment/change phase is thus critical. This order is computed from a *Derivation Graph*, a directed graph

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

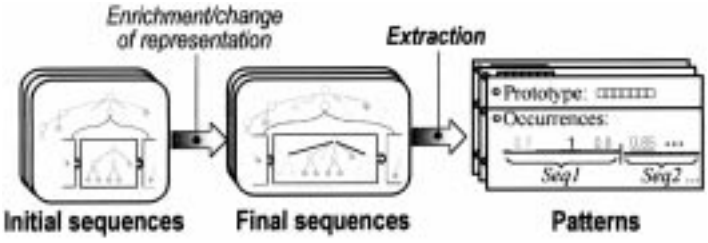


Fig. 2. The enrichment/change of representation within the pattern discovery process.

whose edges each represents a 'description1 → description2' precedence constraint. In Improlog, the Derivation Graph is computed by the system by examining individual derivational procedures, and the user can open a graphical view on it. One possible order for adding the different descriptions chosen by the user is then computed. In some cases, no such possible order exists, which will be signaled by the system. This means that there is incoherence in the derivation structure, i.e., some description D derives from some description D' which itself derives from D. This materializes by Improlog's detecting a cycle in the precedence graph.

One important axle in our description classification schemes distinguishes between *individual*, *local*, and *global*. These three categories of descriptions will be explained below and each illustrated through a (very partial) list of description examples.

- A global description concerns one of the corpus' melodic sequences as a whole. Global descriptions include:
  - ***time signature/meter***
  - ***key signature/overall tonality*** and ***mode***
  - ***harmonic scheme*** (the *chord changes* in the case of tonal jazz)
  - ***tempo***
  - ***pitch histogram, duration histogram***, etc.
  - ***average pitch, average duration***, etc.
- A local description concerns a particular passage (sequence segment):
  - characteristic pitch schemes: ***monotonous contour*** (***ascending***, ***descending***, ***flat*** — i.e., repeated notes), ***local pitch extremum contour*** (ascending contour immediately followed by a descending one, or the reverse — see the section on experimental results), ***gap-fill*** (Narmour, 1989) (p. 220—224), ***arpeggio***.
  - characteristic harmonic schemes: ***cadence***, jazz-type harmonic cells such ***II-V***, or ***VI-II-V-I***
  - grouping structure: all grouping (and boundary) information is handled via local descriptions which possibly overlap. These include ***phrase*** and ***section*** (e.g., to distinguish between A and B sections in a 32AABA-type jazz tune)
  - ***crescendo, decrescendo*** — as determined from the initial representation of melodies when it includes dynamics
  - etc.
- An individual description concerns a particular note (or possibly rest):
  - ***absolute pitch***
  - ***backward interval***, i.e., the interval between a note and its predecessor, expressed either in ***chromatic*** or ***diatonic*** form; ***forward interval***, i.e., the interval between a note and its successor. Having forward and backward intervals simultaneously allows the avoidance of the classical problems associated with the representation of notes by intervals, which intrinsically represent note *pairs* (see, e.g., Cambouropoulos et al., 1999).
  - ***backward interval direction***, i.e., the direction of the backward interval (up/down/repeat), and ***forward interval direction***.
  - (Obviously all pitch variational descriptions make the representation invariant to transposition — either chromatic or diatonic, depending on the particular description considered.)
  - ***degree in overall tonality*** (the tonality of the

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

melody involved); ***degree in local tonality*** (see below)
- ***duration*** (***relative***, i.e., expressed in beats or in very specific contexts; ***absolute***, i.e., expressed in seconds)
- ***duration ratio***, i.e., the quotient between a note's duration and that of its predecessor. This description makes the representation invariant to local distortions of the time axle
- ***duration variation direction*** (lengthened, shortened, or repeated duration)
- ***metric level***: ***binary***, ***ternary***... Taking all of them into account simultaneously allows the handling of, e.g., cases of local poly- or cross-rhythms
- ***metric variation***, and ***metric variation direction***. These track the amount (resp. direction) of metric movement, from weaker beats to stronger ones or vice versa
- ***amplitude*** (dynamics)
- etc. ...

Introducing local and global properties is what allows us to take the system's 'musical sense' beyond the limitations of the note-by-note paradigm. That paradigm, which has prevailed in previous approaches for melodic comparison, has induced major limitations in these approaches, as remarked, e.g., in Widmer (1994). The distinction between a descriptions' horizontal spans has a direct impact on the derivational algorithms that compute the final representation of melodies from their initial representation. There are specific *interaction schemes* between the various descriptions. For instance, the individual description "degreeInOverallTonality" is computed from the global description "overallTonality" and the individual description "absolutePitch". Similarly, the individual description "degreeInLocalTonality" is computed from the local description "localTonality" and the individual description "absolutePitch", where the local tonality may be, for instance, that of a harmonic cell such as a II-V-I in a jazz context. Thus, in a {Cmin7-F7-Bbmaj7} harmonic cell — a major II-V-I in the tonality of Bb — an Eb would get *4* as value of degreeInLocalTonality.

The individual description Salience has a special status. It has a multiple derivational structure (i.e., a dense sub-graph in the Derivation Graph). In fact, because the atypicality of a note along *any* dimension (e.g., a very long duration) generally contributes positively to its salience, Salience derives from all involved individual descriptions. The salience plays a major role in the evaluation of deletions and insertions. A deleted note, whether in a Deletion or in a MultiDeletion, contributes all the more negatively to the similarity as its salience is high (similarly for inserted notes).

OVERALL PRESENTATION OF FlExPat

FlExPat — Flexible Extraction of Patterns — is a general, combinatorial, algorithm for extracting sequential patterns from sequences of data. (To be totally accurate, we should speak of an algorithmic *family* as, for instance, the extraction algorithms we designed take slightly different forms depending on whether the corpus is made of *one* or *several* sequences.)

Algorithmic phases

FlExPat is structured in two main algorithmic phases (see Fig. 3):

- The *passage pair comparison* phase identifies in a computationally economic fashion all equipollent passage couples. A graph, which we call the *similarity graph*, is produced whose vertices each represents a distinct passage and whose edges each represents an equipollent relation

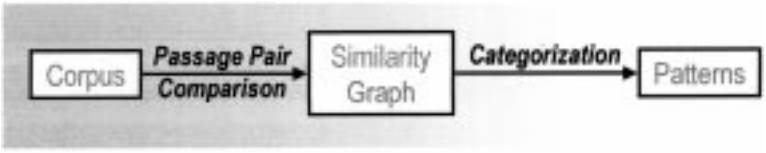


Fig. 3. FlExPat's overall algorithmic scheme.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

between two passages. This graph is generally valued, differentiating edges according to their respective similarity strength.

· The *categorization* phase extracts the actual patterns from the similarity graph. Among possible extraction paradigms, we implemented a method called "star center" which yields very good results while maintaining a fairly low temporal complexity (see below).

Passage comparison

In terms of the passage comparison model, FlExPat allows the use of the Multi-Description Valued Edit Model (**MVEM**) proposed by Rolland and Ganascia (1996b). Among similarity models that have been designed and implemented for comparing melodic passages, many demand that both passages have the same length[3] (see, e.g., Cope, 1991). While this may be appropriate for pattern extraction in very particular types of musical material, it has proven to be a strong limitation in the general case (Rolland and Ganascia, 1999). The MVEM does not impose this equal length constraint, which in particular allows it to handle such fundamental musical devices as ornamentation.

The MVEM specializes the basic *edit model* of which the well-known *edit distance* (Levenshtein, 1965; Needleman and Wunsch, 1970) is an instance. Although the MVEM is a general model for comparing sequential entities of any nature, we will only present it here in the specific context of melodic passage comparison. Similarly, it can be used as a similarity model as well as a dissimilarity model; for the purposes of the MVEM's usage in FlExPat, we will only consider the *similarity* assessment paradigm.

To compare two sequential entities, the optimal correspondence scheme between their respective elements (called *alignment*) is determined (in contrast, in the classical, equal-length comparison models mentioned previously, only one correspondence scheme is considered). An alignment is an ordered series of *pairings,* where a pairing designates the putting in correspondence of a contiguous group of elements of the first entity with a contiguous group of elements of the second entity. One or both groups may contain one or zero elements, as will be shown later on. This paradigm is particularly fit for comparing melodic sequences, as it neatly accounts for such important notions as ornamentation or variation. Any particular instance of the MVEM is characterized by the set of allowed pairing types (APTS), the standard set being $\mathbf{APTS_{standard}}$ = {Insertion, Deletion, Replacement}. Insertion and Deletion are dual pairing types, which correspond respectively to the addition of one note at a determined position in the second passage and to the removal of one note at a determined position in the first passage. Replacement, which corresponds to the putting in correspondence of one note in the first passage with one note in the second, is an abstract type with two actual subtypes:

· Identity, in which the two notes are equal;

· Substitution, where the two notes are different.

By equal, it is meant that, *in the particular musical representation that is currently being used*, the two notes are indistinguishable one from the other. For example, suppose the representation uses the two descriptions "absolute pitch" and "absolute duration" (MIDI-like representation). Then, in Figure 4, the G flat in the first passage is equal to the G

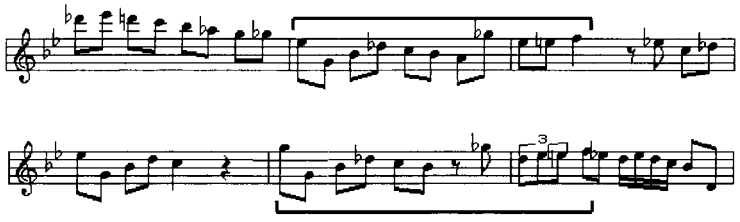


Fig. 4. Two passages of improvised jazz (delimited by horizontal bars above and under staves).

[3]The *length* of a passage is by definition its number of notes (and rests, if the user has chosen that these should explicitly be taken into account.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

flat in the second passage. If the representation uses "backward interval" instead of absolute pitch, then these two notes are different.

For example, let us consider the two improvised jazz passages delimited by horizontal bars above and below the staves in Figure 4. We will assume that the MIDI-like representation noted above is being used.

A possible alignment of the two passages, whose notes[4] have been designated by N1…N11 and N'1…N'11, respectively, is the series of 12 pairings shown in Figure 5.

The first 6 pairings are of type Replacement. The first one is actually an instance of type Substitution, while the following 5 are identities. Then follows a pairing that is a deletion (the removal of the A). Pairing 8 is another identity, pairing 9 is an insertion (the adding of the D), and so on. A more intuitive rendering of the alignment, called a *generalized trace*, is shown in Figure 6. We designed specific graphical representations for instances of every pairing type. A cross over a note means its deletion, while a capitalized lambda above a note means its insertion. Replacements are denoted by triple lines (which is meant as a reminder of the specific equality symbol ≡); continuous lines are used for identities, dashed lines for substitutions. Differences between notes involved in substitutions are of two separate natures here:

- Difference in pitch, as in the first substitution
- Difference in duration, as in the last three.

$$\begin{pmatrix} N1 \\ N'1 \end{pmatrix}\begin{pmatrix} N2 \\ N'2 \end{pmatrix}\begin{pmatrix} N3 \\ N'3 \end{pmatrix}\begin{pmatrix} N4 \\ N'4 \end{pmatrix}\begin{pmatrix} N5 \\ N'5 \end{pmatrix}\begin{pmatrix} N6 \\ N'6 \end{pmatrix}\begin{pmatrix} N7 \\ - \end{pmatrix}\begin{pmatrix} N8 \\ N'7 \end{pmatrix}\begin{pmatrix} - \\ N'8 \end{pmatrix}\begin{pmatrix} N9 \\ N'9 \end{pmatrix}\begin{pmatrix} N10 \\ N'10 \end{pmatrix}\begin{pmatrix} N11 \\ N'11 \end{pmatrix}$$

Fig. 5. A possible alignment between the two passages.

(Obviously, a substitution can also involve differences in *both* descriptions.)

Non-standard pairing types include the following (Rolland and Ganascia, 1999):

- *MultiDeletion*, i.e., the deletion of a whole (contiguous) group of notes; and its dual type, *MultiInsertion;*
- *Fragmentation* (Mongeau and Sankoff, 1990), i.e., the putting in correspondence of one note in the first passage and two or more notes in the second; an example is shown in Figure 7. The upper note is fragmented into the lower group (this triple attack is a technical/stylistic device frequently used by Charlie Parker).
- *Consolidation*, the type which is the dual of Fragmentation, i.e., the putting in correspondence of two or more notes in the first passage and one note in the second;
- Swap, corresponding to the inversion of two successive notes;
- *MultiReplacement*, i.e., the putting in correspondence of a contiguous group of two or more notes in the first passage and a contiguous group of two or more notes in the second.

As a global illustration, let us assume that the APTS is:

APTS$_1$ = {Insertion, Deletion, Replacement, MultiInsertion, MultiDeletion, MultiReplacement}

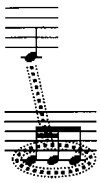


Fig. 7. Fragmentation example.

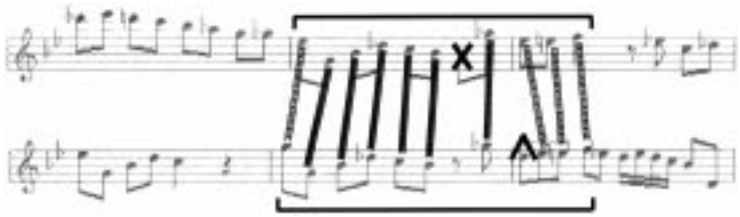


Fig. 6. One possible alignment between the two passages, using the standard Allowed Pairing Type Set: APTS$_{standard}$ = {Insertion, Deletion, Replacement}.

[4]In this example, rests are ignored. In the Improlog system (see the section on implementation) rests are generally treated as particular, degenerated, notes which allows for more meaningful melodic comparison metrics.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

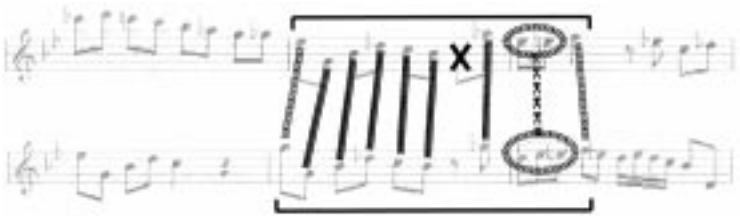


Fig. 8. Another alignment between the two passages, using APTS = {Insertion, Deletion, Replacement, MultiInsertion, MultiDeletion, MultiReplacement}.

Figure 8 shows another possible alignment, an alternative to that of Figure 6, between the same two passages assuming that a multiReplacement is used for the putting in correspondence of the note group located in the last bar of each passage. This may be considered more conform to musical intuition than using two substitutions and a deletion as done previously. In fact, a multiReplacement captures more directly the concept of the melodic variation/ornamentation involved.

If **P** is an example of a pairing type, we call the *signature of* P the integer couple containing the respective lengths of the two segments put in correspondence. One of the groups may be empty (length 0), as for insertions and deletions, and one or both groups may contain only one note (length 1), as for replacements, insertions, and deletions. Thus, Replacement instances have signature {1;1}, Swap instances {2;2}, Insertion instances {0;1}, and Deletion instances {1;0}. The signature of MultiInsertion, MultiDeletion, Fragmentation, Consolidation, and MultiReplacement instances varies as a function of the particular instance involved. For example, the Fragmentation instance shown in Figure 7 has signature {1;3}, while the MutiReplacement instance has signature {2;3}. We thus distinguish between pairing types that have a *fixed* signature (Replacement, Insertion, Deletion, Swap) and pairing types that have a *variable* one (Fragmentation, Consolidation, MultiInsertion, MultiDeletion, and MultiReplacement). For every variable pairing type, the range of possible signatures for instances is stated explicitly. For example, for a particular instance of the MVEM whose APTS contains MultiInsertion (resp. MultiDeletion), the possible signatures allowed for multiInsertions may be stated as the set Sig_1 = {{0;2}, {0;3}, {0;4}} (resp. Sig_2 = {{2;0}, {3;0}, {4;0}}). Similarly, for that MVEM instance, the set of allowed signatures for MultiReplacement may be Sig_3 = {{2;3}, {3;3}, {3;2}, {3;3}}. The important notion of *effective size,* which will be used in the rest of this article, can now be defined. While the *size* of an APTS is simply the number of pairing types it contains, its effective size, denoted by $\|\mathbf{APTS}\|$, is equal to the number of all possible signatures of instances of all the pairing types in the APTS. When the APTS only contains fixed pairing types, its effective size is equal to its cardinality, e.g., 3 for $APTS_{standard}$. Otherwise, the effective size is always greater than the cardinality; for example, $APTS_1$ has cardinality 6 and effective size 13 (=1+1+1+3+3+4) when using the Sig_1, Sig_2, and Sig_3 signature sets.

We will now explain what the 'valued' adjective in the model's name means. In the MVEM, a *contribution function* is associated with each pairing type, so any pairing in an alignment gets a numerical evaluation reflecting its individual contribution to the overall similarity. Contributions can be positive, as is always the case for identities, or negative, as is the case for insertions and deletions. The *value* of an alignment is then defined as the sum of the contributions of all of the alignment's pairings. We compute the similarity between two passages as the highest possible value of any alignment between them. In other words, the similarity between passages π and π' is the value of the best alignment between π and π' (or that of the best ex-aequo alignments where appropriate).

The various melody/note descriptions are simultaneously taken into account in contribution functions using a weighted linear combination paradigm. For example, depending on the context, the

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

deletion of a note with a long duration and a strong metric position will have a more negative contribution than the deletion of a less prominent note. We have designed and implemented appropriate contribution functions for all pairing types and all musical descriptions taken into account in Imprology. These descriptions include, of course, the derived ones automatically added by the system during the representation enrichment/change phase. The possibility, for the user, to dynamically choose the set of descriptions used and their respective weights makes possible the adoption of different *viewpoints* on melodic similarity and, hence, on the nature of extracted patterns. For instance, the user may at some point privilege temporal descriptions (durations, metrics, etc.) w.r.t. frequential descriptions (pitches, intervals, etc.) to achieve a more rhythmically oriented pattern discovery. We have also proposed and implemented methods for automatically optimizing description weights (Rolland and Ganascia, 1996a).

Although any range for similarity values may be considered, it is convenient to used a *normalized similarity range* where similarity values are in the [0,1] interval. A similarity of 1 means that the passages are equal.

Algorithms (e.g., Needleman and Wunsch, 1970) based on dynamic programming have been designed for computing the optimal alignment between two sequences, i.e., the one with the highest value. For comparing passages π, with length m, and π', with length m', these algorithms take time (and memory) that is proportional to $m.m'.||APTS||$. In FlExPat, to compute the similarity between two passages, instead of computing it from scratch, the results of the previous comparison of different pairs of shorter passages are reused. This results in a dramatic reduction of computation time, as will be described in the subsection on passage pair comparison phase.

MORE DETAILED DESCRIPTION OF FlExPat

We will now present FlExPat's key characteristics and properties whose detailed descriptions and/or proof are given in Rolland (1998).

Input

FlExPat's main inputs are:

- a sequence corpus (possibly made of only one sequence)
- integers $\boldsymbol{m_{min}}$ and $\boldsymbol{m_{max}}$ controlling the minimum (resp. maximum) length of passages constitutive of patterns. The action of these parameters may be neutralized by setting them to extreme values: m_{min} set to 1 and m_{max} set to the corpus' cumulative length (the sum of the lengths of all sequences). However, in practice, extremely short patterns, such as single notes, are very seldom meaningful. Similarly, extremely long patterns are very rare: think of a recurrent phrase of 100 notes, for instance, especially if the sequences in which it is sought have length 200 or so
- an integer function $\Delta\boldsymbol{m}$ controlling the maximum possible length difference between compared passages. It is generally appropriate to define Δm as a function of the length m of the passages under consideration, which is expressing the maximum allowed length difference as a *percentage* of the passages' length. For instance, one may choose:

$$\Delta m(m) = \left\lfloor \frac{m}{3} \right\rfloor$$

- In that case, a passage of length m = 4 may only be equipollence-tested to passages of length 3, 4, and 5 (a maximum of one note addition/removal). A passage of length 10 may only be equipollence-tested to passages whose lengths are between 7 and 13. Here again, the action of Δm may be neutralized by setting $\Delta m \equiv \infty$. However, such a setting rarely has any musical meaning — when would one want to consider significantly similar a two-note passage and a 25-note one? Furthermore, we have established theoretically that there is always a finite value D such that, for any constant $C \geqslant D$, setting $\Delta m \equiv C$ is equivalent to setting $\Delta m \equiv \infty$.
- a *fully specified instance of the MVEM*: description set and weights, set APTS of allowed pairing types along with associated contribution functions;
- a *similarity threshold* defining equipollence

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

between passages. Typical appropriate thresholds when using a normalized similarity scale are 0.75 or 0.8.

Passage Pair Comparison Phase

First, the corpus' sequences are concatenated in order, yielding a *global sequence* **S** of length **L**. After an initialization phase which includes the creation of an 'empty' similarity graph, the space of all candidate passage pairs is explored in a specific order, viz. increasing passage positions combined with increasing passage lengths (see below). For each candidate pair of passages, their similarity value is computed, and ***if it is above the threshold, an edge is created*** between the corresponding two vertices in the similarity graph, denoting equipollence. Several of the above terms will now be given precise definitions:

- We call *position* of a passage, the index **i** in S of its first note (or rest, if the user has chosen that these should be explicitly taken into account).
- The *length* **m** of a passage is, as stated previously, its number of notes (and rests, if appropriate).
- A *candidate* passage pair is, by definition, one that satisfies the constraints stated above:
  - both segments must be unmixed, i.e., must not be spread over the boundary between two successive sequences.
  - their respective lengths must be between m_{min} and m_{max}
  - their length difference must be small
  - other (optional) constraints should also be satisfied concerning the size of the possible overlap between the two passages. For simplicity, these overlap constraints are not detailed here.

Any passage is uniquely defined in the corpus, i.e., in S, by the **(i,m)** couple. During the passage pair comparison phase, the passages π' = (i',m') that will be compared to π = (i,m) are such that i' $\leqslant$ i (which, among others, guarantees that no two passages may be compared more than once). More precisely, passages (1,m_{min}), then (1,m_{min}+1),then (1,m_{max}), then (2,m_{min}), then (2,m_{max}), then (L-m_{max}+1,m_{max}) will be processed in order. *Processing* a given passage π = (i,m) means computing its similarity and testing its equipollence with the following passages in order: (1,m$_{min}$), (1,m_{min}+1), (1,m_{max}), (2,m_{min}), ... (i-1,m_{max}), (i,1), (i,m-1). Of course, this particular list differs when π is located near a sequence boundary.

The ***core property*** in FlExPat's passage pair comparison algorithm, which is what annihilates combinatorial explosion, derives from the recurrence relation shown in Equation 1. For clarity, we will assume here that the set of allowed pairing types is the standard set. This relation expresses the similarity, denoted, $\mathrm{Sim}_{i'm'}^{im}$ between passages π and π' — of respective positions i and i' in the corpus (i.e., in global sequence S) and respective lengths m and m' — as a function of the similarity values between three other passage pairs. Each passage in those pairs is either π or π', or *prefixes* of π or π' of length m-1 (resp. m'-1), i.e., the results of the removal of the last element of π or π'. Because of the specific exploration order of the passage pair space, it is guaranteed that all three similarity values have already been computed (and stored). The aforementioned core property can thus be stated:

The computation of the similarity between any (new) pair of candidate passages takes constant time.

Designating the number of maximization lines in Equation 1, this computation time is the sum of times required for accessing nl values of the Sim matrix, for accessing nl values of the *contrib* matrix, for making nl additions, and for finding the greatest of nl numbers. It is, therefore, proportional to nl. However, nl is clearly the effective number of different pairings, i.e., the *effective size* of APTS (cf. the subsection on passage comparison). In summary, the constant time required for computing any similarity value $\mathrm{Sim}_{i'm'}^{im}$ is proportional to $\|\mathrm{APTS}\|^2$. This required time would be $O(m.m'.\|\mathrm{APTS}\|^2)$ if efficient sequence comparison algorithms (e.g., Needleman and Wunsch, 1970) were used. In other words, it would be non-constant since it would depend on the length of the passages compared. Compared to FlExPat, the time required would be multiplied by a factor $m \times m'$. As a total, a dramatic enhancement of algorithmic efficiency is achieved over the whole passage pair comparison phase (see the subsection on algorithmic complexity). This achievement is due to the specific Dynamic Programming tech-

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

niques we designed, which benefit from the recurrence relation in Equation 1.

$$Sim^{i}_{i'}{}^{m}_{m'} = \max\left\{\begin{array}{ll} contrib\begin{pmatrix} S[i+m-1] \\ S[i'+m'-1] \end{pmatrix} & + \; Sim^{i}_{i'}{}^{m-1}_{m'-1} \\ contrib\begin{pmatrix} S[i+m-1] \\ - \end{pmatrix} & + \; Sim^{i}_{i'}{}^{m-1}_{m'} \\ contrib\begin{pmatrix} - \\ S[i'+m'-1] \end{pmatrix} & + \; Sim^{i}_{i'}{}^{m}_{m'-1} \end{array}\right. \quad (1)$$

Initialization of the 4-dimensional similarity matrix *Sim* is carried out following the operations depicted in Figure 9. For m' = 0 (second line in figure), the value of Sim is in fact independent of i' and thus denoted Sim^{im}_{0}. For any given i, the values of Sim^{im}_{0} can all be computed in increasing values of m, by using the recurrence relation of line 2, starting from root value $Sim^{im}_{i'0}$ which is 0 for all i, i'. Similarly, for any given i', the values of $Sim^{im}_{i'0}$, noted $Sim^{0}_{i'm'}$, can all be computed in increasing values of m', by using the recurrence relation of line 3.

Categorization Phase

Among possible approaches for extracting patterns from the similarity graph, we have proposed the Star Center algorithm, which is influenced by an approach proposed in molecular biology (Gusfield, 1993). Figure 10 shows an example of such a star, with numbers (edge values) each denoting a prototype-occurrence similarity value. The extracted pattern's prototype "is" the star center, while its occurrences "are" the ends of the star's branches. The corresponding algorithm can be sketched as follows:

1. For each vertex v in the similarity graph, ***compute***:

$$totalValuation(v) = \sum_{v' \in adj(v)} value(v, v')$$

2. ***Sort*** the set of stars by decreasing totalValuation.

This yields a list of patterns (stars) ordered in decreasing *prominence*, the prominence of a pattern being naturally defined as its corresponding totalValuation. For instance, the prominence of the star shown in Figure 10 is 4.4 (=1+.9+.9+.83+.77).

To handle possible cases of ex æquo, we have proposed, implemented, and experimentally validated additional sorting strategies, which are not detailed in this article. We have also proposed various online/offline ***filtering*** approaches for fitting the above-mentioned possible additional pattern constraints. For example, patterns whose quorum (cf. the section on formalization) is below a fixed threshold q can be filtered out from FlExPat's output. This is straightforwardly carried out by sequentially traversing the set of extracted stars and, for each, computing its quorum and comparing it to q. Computing a star's quorum takes time which is proportional to its size, so the filtering of the whole of FlExPat's output only takes fractions of seconds in practice. (An auxiliary table is precomputed which provides 'note index → sequence number in corpus' information; thus, finding the sequence number associated to any vertex takes constant time: one access to the table).

Specific advantages of the Star Center approach are the following:

- it directly and naturally yields a prototype for every pattern (the passage corresponding to the Star's central vertex). "Directly" means that no

$$Sim^{i}_{i'}{}^{0}_{0} = 0$$

$$Sim^{i}_{i'}{}^{m}_{0} \longrightarrow Sim^{i}_{0}{}^{m} = contrib\begin{pmatrix} S[i+m-1] \\ - \end{pmatrix} + Sim^{i}_{0}{}^{m-1}$$

$$Sim^{i}_{i'}{}^{0}_{m'} \longrightarrow Sim^{i}_{i'}{}^{0}_{m'} = contrib\begin{pmatrix} - \\ S[i'+m'-1] \end{pmatrix} + Sim^{i}_{i'}{}^{0}_{m'-1}$$

Fig. 9. Initialization operations.

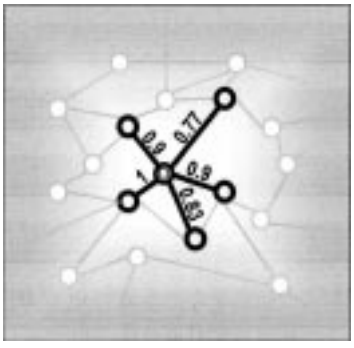


Fig. 10. Example ∗ in the Similarity Graph.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

additional computing is needed after the patterns have been extracted. "Naturally" refers to the fact that the prototype plays an explicit role in the extraction of a pattern, as opposed to approaches where prototypes are computed 'off-line' from already extracted sets of occurrences.
- a quantitative prominence evaluation is obtained for every pattern, once again without any additional computation;
- its algorithmic complexity is satisfactory. In particular, its time complexity (see also the section Discussion and related work) is proportional to the number of edges in the similarity graph, which is (loosely) bound by $L^2 (m_{max} - m_{min} + 1)^2$. This is to be compared to other possible approaches for extracting patterns from the graph; for instance, finding [maximal] cliques is known to be an NP-complete problem (Aho et al., 1974).

Algorithmic Complexity

For the cumulated complexity of FlExPat's two phases, the following (loose) upper bounds can be given.

- ***Time complexity***: $O(L^2 {m_{max}}^2 ||APTS||)$.
- ***Space complexity***: $O(L^2 {m_{max}}^2)$, the dominant term in the sum being the memory required for storing the similarity graph.

In practice, for example in musical contexts, m_{min} and m_{max} are always set to small, constant values with respect to L, so that both complexity bounds merely rewrite to $\mathbf{O(L^2)}$.

The above are theoretical bounds. They can only be reached in totally extreme and meaningless situations where:

- any pair of passages would always be equipollent
- all pertinent combinatoric-controlling devices such as the Δm function would be neutralized. This would lead to a *complete* similarity graph which:
- contains vertices corresponding to *all* possible passages of the corpus (with any length and position)
- has an equipollence edge between *any* couple of vertices.

Practically, this is a totally unrealistic situation: measurements actually obtained on real (musical) data are thousands or millions of times smaller — see, e.g., the measured graph dimensions in the subsection Experimental results, discussed below.

IMPLEMENTATION AND EXPERIMENTAL RESULTS

Improlology

Improlology (Rolland, 1998; Rolland and Ganascia, 1996a) is an open environment which we designed and implemented to carry out various music-related tasks, including pattern extraction and localization as well as content-based information retrieval from music databases (Rolland et al., 1999). The addition of a significant set of different musical descriptions, as requested by the user or the system, is carried out automatically. Automated Pattern extraction algorithms currently implemented in Improlology are the following:

- FlExPat;
- KMR (Knuth et al., 1977), a fast algorithm for finding *exact* repetitions (fixed-length, equality-based passage comparison model), that we have extended to deal with a corpus of sequences instead of a single sequence.

The system's name, literally meaning 'study of improvisation', echoes a naming tradition that was significant at some point in jazz history. At that time, many songs were named with an "ology" suffix, for instance *Anthropology*, *Crazeology*, *Bopology*, and, of course, Parker's *Ornithology*. Improlology's link with jazz stems in particular from the various experiments that have been conducted on improvised jazz material.

Improlology is implemented in the Smalltalk-80 object-oriented language. Altogether, it represents several hundred *classes* and *methods*. It reuses, adapts, and extends the MusES collection of Smalltalk *classes* and *methods* (Pachet, 1994) for representing and manipulating basic tonal music concepts. Various visualization and graphical edition tools are offered to the user, including score editors, chord sequence/grid editors, a derivation graph viewer, and a similarity graph viewer. Data format converters have also been implemented, e.g., to import and export Standard MIDI Files (SMF).

The implementation (dialect) of the Smalltalk

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

used is ParcPlace-Digitalk's VisualWorks. The latter's total portability allows Improlog to be able to run under MS Windows, MacOS, Unix, and Linux.

The system is *open* in the following sense: let us consider the representation enrichment/change phase. Beyond additional descriptions that the system can automatically compute, the user may 'manually' add other descriptions (global, local, and/or individual) to the melodies under study. (The user may also, of course, add the Smalltalk code in order to automatically compute these new descriptions). S/he then has to design contribution functions that are appropriate with respect to the newly introduced descriptions. Similar observations apply for the addition of new pairing types by the user. The choice of an object-oriented programming environment such as ParcPlace-Digitalk's VisualWorks makes these different user processes easier.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

Experimental results

Input and parameter settings

In order to give a few examples of results, an example corpus will be considered. This corpus, which will be referred to as **C1**, is made of 10 Parker solos in the 'C major - Blues' category of Owens' corpus:

- 3 takes[5] on "Cool Blues"
- 3 takes on "Relaxin' at Camarillo"
- 4 takes on "Perhaps"

C1's cumulative length L is about 2,000 (notes and rests). Each solo (except one) comprises between 25 and 40 bars and between 130 and 250 notes.

The description set used is the following (as introduced in the subsection Music representation):

- Global descriptions: ***time signature*** (of each sequence)
- Local descriptions: ***local pitch extremum***
- Individual descriptions:
  - ***Duration***, expressed in relative time (beats)
  - ***Backward and forward chromatic intervals***
  - ***Binary and Ternary metric levels***
  - ***Position in local pitch extremum.*** This allows us, in particular, to distinguish the peculiar status of peak notes (especially local pitch maxima). Indeed, these notes often have a perceived (and intended) prominence, as has been evidenced in improvised jazz (Owens, 1974). They often have a strong contribution (positive or negative) to the similarity perceived between two passages.
- Additionally, rests are represented explicitly, as particular (degenerated) kinds of notes.

Among FlExPat's parameter values are the following :

- $m_{min} = \mathbf{3}$;
- $m_{max} = \mathbf{27}$;
- Equipollence defined by a **0.7** threshold on normalized similarity values;
- $\Delta m(m) = \lfloor \frac{m}{3} \rfloor$
- Characteristics of the MVEM instance used (the contribution functions associated with the different pairing types will not be detailed here):
  - APTS = $\mathbf{APTS_{standard}}$
  - Respective weights for the different pairing types: **2.5** for deletion, **2.5** for insertion, **1** for replacement
  - Respective weights for the different descriptions: **1** for 'Duration', **1** for 'ChromaticIntervals', **0.5** for 'MetricalLevels', **1** for 'PositionInLocalPitchExtremum'

Output

In quantitative terms, the results of the pattern extraction are the following:

- The similarity graph has about 12,000 vertices and 40,000 edges. These should be compared to the maximal graph dimensions as stated theoretically (see the subsection Algorithmic complexity): about 50,000 vertices (L*[mmax-mmin+1] different passages) and about 1,250,000,000 edges (1/2*V*[V+1] where V is the number of vertices).
- The largest star has a size of 35 (i.e., 34 edges + the center).
- The longest pattern has a prototype of length 27 (the length of a pattern being defined as the length of its prototype or as the average length of its occurrences).

[5]Each *take* is a different recording, i.e., a different improvisation, of the same *tune* (e.g., Cool Blues) during the same studio session.

Fig. 11. Prototype of pattern Pat1:[19:3-23:1] passage of *Relaxin' at Camarillo Take 4* solo.

Fig. 12. First occurrence of Pat1: [7:4-11:1] passage of *Relaxin' at Camarillo Take 4* solo.

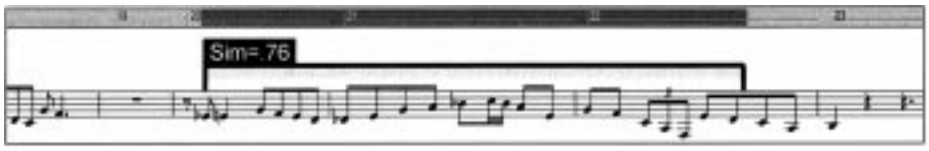


Fig. 13. Second occurrence of Pat1 [32:1.75-35:1] passage of *Perhaps Take 3* solo.

Fig. 14. Third occurrence of Pat1: [20:1.5-22:4] passage of *Relaxin' at Camarillo Take 3* solo.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

- The maximum possible quorum, i.e., 10, is reached for 58 patterns — this means that patterns have been found whose occurrences/prototype are spread out in all ten of the corpus' sequences. These patterns have a length less than or equal to 4 and possess at most 21 occurrences.

In qualitative terms: since it would be totally inappropriate to fill pages and pages of this article with scores showing examples of extracted pattern prototypes and occurrences, we will focus instead on one pattern. Figure 11 shows the prototype (length 26) of a pattern extracted from C1 by Improlog/FlExPat, which we will refer to as **Pat1**. It is among the group of the longest extracted patterns, and is the best of those with length 26 — i.e., it has the highest totalValuation. The next three figures show its 3 occurrences throughout C1, ordered by decreasing computed similarity with the prototype. Respective similarity values are 0.93, 0.76, and 0.75. The prototype's normalized auto-similarity (i.e., its similarity with itself) is by definition equal to 1. Each passage is delimited by the usual ‹bar No.›:‹beat No.› notation in captions and visually by the horizontal bracket on top of each staff. The grey scale of the bar reflects the similarity value, ranging continuously from black (Sim = 1: equality) to white (Sim = 0.7, the value of the threshold).

This pattern does make musical sense and shows the system's similarity assessment qualities, which are due to the MVEM's characteristics. The prototype and occurrences all 'share' a common final segment, viz. the [20:3.5-23:1] passage of the prototype. The initial segment is different for each of them, but similar — in particular, all four initial segments end with a G, of varying duration and metric placement. The putting in correspondence by the system of two particular sub-segments appears particularly appropriate: the diatonic descents before the first C in the prototype and first occurrence (F-Eb-D and D-C-B, respectively) are matched, despite a local transposition. Importantly, the initial (variable) segments in Pat1's prototype

and occurrences have different lengths (7, 7, 3, 4, respectively). As a result, all pattern extraction algorithms based on an equal-length passage comparison model (cf. subsection: Passage comparison) ***fail*** to extract Pat1. Instead, they extract at most the reduced, incomplete pattern (fixed part).

Of course, this is no more than one particular example of a pattern, with clear musical pertinence, among the ones extracted by Improlog/FlExPat from such a corpus. Nevertheless, this does illustrate several key points, which will be discussed in the next section. In particular, Pat1 was not signaled by Owens, thus being effectively 'discovered' by the system. One plausible reason is that Owens studied the corpus as a whole and could not afford, in an already years-long study, to consider different sub-corpora separately. Using FlExPat, patterns *local* to C1 have been discovered, i.e., patterns such as Pat1 which only appear in Parker's solos over C major Blues chord changes.

Among extracted patterns are also a number of patterns that are part of Owens' lexicon. Figure 15 shows three such patterns of various lengths and nature; for example, the second one is a mere 8^{th}-note chromatic descent. The codes above the staffs are pattern codes as classified in Owens' lexicon.

DISCUSSION AND RELATED WORK

Overall, based on the whole set of results we obtained with different (sub-)corpora and different description and description-weighting sets, we have made the following key observations :

- A large number of Owens' patterns were satisfactorily extracted by the system, which conforms to our validation objective. Even more interestingly, a number of new and musically meaningful patterns, such as Pat1 described above, were also 'discovered' by the system. The total automation of the pattern extraction process makes it possible, for a music analyst/musicologist, to discover specific patterns from any arbitrary subset of a given corpus. One example was seen previously through pattern Pat1, which was extracted from the C1 sub-corpus.
- Significantly different sets of melody/note descriptions or description weights — i.e., different viewpoints w.r.t. musical similarity — lead to the extraction of significantly different patterns. In particular, the use of too-restrictive sets leads to the extraction of patterns without any musical pertinence.
- Extracted patterns are very often specific to particular melodies (sequences) of the corpus or to related groups of melodies. For instance, the best occurrence of Pat1 above is *internal* (same tune and take as the prototype), and 3 out of the 4 passages belong to the same tune (*Relaxin' at Camarillo*); occurrence profiles observed with the above-mentioned Owens' lexicon pattern display even clearer trends in this respect. This presents a clear perspective in automated pattern-based identification of style or author.

In addition, the division of FlExPat in two algorithmic phases appears as a very rich feature. Indeed, it allows the user to experiment with various categorization strategies and/or parameter settings (e.g., combining various filtering options in the Star Center algorithm) once the similarity graph has been generated by the more computationally expensive passage pair comparison phase. The very short running times of the Star Center algorithm allow this to be made in the interactive mode. Table 1 gives example of measured running times, in seconds, for this algorithm using different global sequence length L values a typical hardware configurations.

Table 1. Experimental measurements of Star Center algorithm's running time patterns (in seconds).

L	50	500	5000
Intel Pentium II 400 MHz (64 MB RAM)	0.003	0.021	0.83

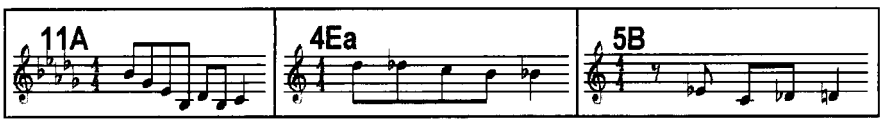


Fig. 15. Examples of patterns as they appear in Owens' lexicon, which were extracted by Improlog/FlExPat from C1.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

The systems whose pattern extraction paradigms are most similar to ours are Cope's EMI (Cambouropoulos et al., 1999) and Pennycook et al.'s system (1993). Due to its real-time functioning capability, the latter skirts around combinatorial aspects inherent to sequential pattern extraction, using an online phrase segmentation algorithm upon which it strongly relies. EMI's pattern extractor's main limitations lie in its major combinatorial and algorithmic redundancy and in its imposed drastic equipollence criteria (Hamming distance). In fact, to our knowledge no general pattern discovery algorithm other than FlExPat allows the use of an edit model with non-constant contribution functions and user- or system-adjustable description set and weights.

FUTURE WORK

As musical pattern discovery is a very general problem, a number of perspectives to this work can be drawn, some of which have already been touched upon in this article. One is the merger of Improlog and the *ImPact* jazz bass accompaniment system, a work that has already been started (Ramalho et al., 1999). ImPact's main source of generative material is a *Musical Memory* (MM) that contains typical melodic fragments that are recurrent in a (human) bass player's improvisations. The MM is indexed by the harmonic schemes typical of tonal jazz (such as II-V-I, as mentioned previously). The integration of FlExPat in Impact allows that both the initial constitution and the incremental enrichment of the MM can now be automatically undertaken by Improlog, as opposed to manually (user). ImPact, whose ultimate goal is to be a *real-time* jazz accompanist, does pattern extraction during its 'spare time' from musical material provided by a human 'tutor'. Between any two performances, it listens to, and learns from, other improvised material (by human performers — and, ultimately, by other computer performers?); this mimics the 'learning by imitation' methodology which is widely recommended, and used by, jazz improvisation 'students'.

Another perspective is the integration of FlExPat within a music information retrieval system. This implemented system, Melodiscov (Rolland et al., 1999), implements the content-based retrieval paradigm which we have dubbed *WYHIWYG* (What You Hum Is What You Get). The user queries a music database (or the World Wide Web) by humming a few notes into a microphone. Alternatively, the user may sing (with lyrics) or whistle, still with excellent results. For every melodic pattern found by FlExPat in the database (and there are, in practice, huge numbers of such patterns in, e.g., pop/rock databases), it may be sufficient to compare the query pattern once to the pattern's prototype instead of comparing it to all of its occurrences. This way, a tremendous efficiency increase w.r.t. existing WYHIWYG algorithms could be obtained via such a (pre)processing of the database before searching.

REFERENCES

Aho, A.V., Hopcroft, J.E., and Ullman, J.D. (1974). *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley.

Anagnostopoulou, C., Hörnel, D., and Höthker, K. (1999). Investigating the influence of representations and algorithms in music classification. In E. Cambouropoulos and P.Y. Rolland (Organizers), *Focus Workshop on Pattern Processing in Music Analysis and Creation*. AISB'99 Convention, Edinburgh, April 6—7, 1999.

Cambouropoulos, E., Crawford, T., and Iliopoulos, C.S. (1999). Pattern processing in melodic sequences: Challenges, caveats and prospects. In E. Cambouropoulos and P.Y. Rolland (Organizers), *Focus Workshop on Pattern Processing in Music Analysis and Creation.* AISB'99 Convention, Edinburgh, April 6—7, 1999.

Cope, D. (1991). *Computers and Musical Style*. Oxford: Oxford University Press.

Gusfield, D. (1993). Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55, 141—154.

Knuth, D.E., Morris Jr., J.H., and Pratt, V.R. (1977). Fast pattern matching in strings. *SIAM J. Comput*. 6, 323—350.

Levenshtein, V.I. (1965). Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory,* 10(8), 707—710.

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

Narmour, E. (1989). *The Analysis and Cognition of Basic Melodic Structures: the Implication-Realization Model.* Chicago: University of Chicago Press.

Needleman, S., and Wunsch, C.D. (1970). A general method applicable to the search for similarities in the amino-acid sequence of two proteins. *Journal of Molecular Biology*, 48, 443—453.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011

Owens, T. (1974). *Charlie Parker: Techniques of Improvisation*. Ph.D. Thesis, Department of Music, University of California at Los Angeles (UCLA).

Pachet, F. (1994). The MusES system: an environment for experimenting with knowledge representation techniques in tonal harmony. In *First Brazilian Symposium on Computer Music — SBC&M '94* (pp. 195—201).

Pennycook, B., Stammen, D.R., and Reynolds, D. (1993). Toward a computer model of a jazz improviser. In *Proceedings of the 1993 International Computer Music Conference* (pp. 228—231).

Ramalho, G., Rolland, P.Y., and Ganascia, J.G. (1999). An artificially intelligent jazz performer. To appear in *Journal of New Music Research*.

Rolland, P.Y. (1998). *Découverte Automatique de Régularités dans les Séquences et Application à l'Analyse Musicale*. Thèse de Doctorat en Informatique de l'Université Paris VI, July 1998.

Rolland, P.Y., and Ganascia, J.G. (1996a). Automated motive oriented analysis of musical corpuses: a jazz case study. In *Proceedings of the 20th International Computer Music Conference, ICMC'96*, Hong Kong (pp. 240—243).

Rolland, P.Y., and Ganascia, J.G. (1996b). Automated identification of prominent motives in jazz solo corpuses. In *Proceedings of the 4th International Conference on Music Perception and Cognition ICMPC'96*, Montreal (pp. 491—495).

Rolland, P.Y., and Ganascia, J.G. (1999). Musical pattern extraction and similarity assessment. In E. Miranda (Ed.), *Readings in Music and Artificial Intelligence*. Harwood Academic Publishers. (To appear)

Rolland, P.Y., Raskinis, G., and Ganascia, J.G. (1999). Musical content-based retrieval: an overview of the Melodiscov approach and system. Seventh ACM International Multimedia Conference, Orlando, November 1999. (To appear.)

Rosch, E. (1978). Principles of categorization. In E. Rosch and B.B. Lloyd (Eds.), *Cognition and Categorization* (pp. 27—48). Laurence Erlbaum Associates.

Rowe, R. (1993). *Interactive Music Systems*. MIT Press.

Suppes, P., Krantz, D.H., Luce, R.D., and Tversky, A. (1990). *Foundations of Measurement*, Vol. 2. New York: Academic Press.

Widmer, G. (1992). Qualitative perception modeling and intelligent musical learning. *Computer Music Journal* 16(2).

Widmer, G. (1994). The synergy of music theory and AI: Learning multi-level expressive interpretation. Extended version of a paper appearing in *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, USA.

Pierre-Yves Rolland
Laboratoire d'Informatique de Paris 6 (LIP6)
CNRS UMR 7606 – Université Pierre et Marie Curie
France
E-mail: p.y_rolland@yahoo.com

Dr. Pierre-Yves Rolland received his PhD in computer science from Université Pierre et Marie Curie-Paris 6 in 1998. He is currently assistant professor in computer science and artificial intelligence at Université d'Aix-marseille III and researcher at LIP6 computer science laboratory, Paris. His research interests include sequential data mining, artificial intelligence, cognitive modeling, and their application to the computational modeling of music analysis and creation. He is also conducting research related to musical content extraction and content-based retrieval in music databases.

Downloaded By: [Aalborg University] At: 14:28 11 May 2011