PATMINR: IN-DEPTH MOTIVIC ANALYSIS OF SYMBOLIC MONOPHONIC SEQUENCES

Olivier Lartillot
Aalborg University, Department of Architecture, Design and Media Technology, Denmark
olartillot@gmail.com

ABSTRACT

A version of *PatMinr* [2] has been submitted to the MIREX task on Discovery of Repeated Themes & Sections [1]. *PatMinr* can find repetitions of sequential patterns from monophonic sequences represented in a symbolic format.

This document presents the model in more details, and specifies the particular parameters used in the version submitted to the MIREX competition.

1. VERSION SUBMITTED TO MIREX

The *MiningSuite* [1] is a new open-source framework for audio and music analysis implemented in Matlab decomposed into modules. In particular, the *MusMinr* module concentrates on music analysis and features an option for motivic analysis. The actual pattern analysis computation is taken care by the *PatMinr* module dedicated to pattern analysis. Version 0.7.1 of the *MiningSuite* has been submitted to MIREX.

1.1 *mus.minr*: pattern discovery

Music sequences in symbolic format can be loaded using the text file format defined in the MIREX task, using the following syntax:

```
mus.minr('filename.txt', 'Motif')
```

where `mus.minr` loads the text file and perform a selection of music analysis methods on each successive note. Here, only the `'Motif'` method is called, corresponding to the motivic analysis, which uses the pattern mining algorithms in the *PatMinr* module of the *MiningSuite*.

In this version, the algorithm searches for all possible repetitions of subsequences along any subset of musical dimensions. The subsequences can be *heterogeneous*, in the sense that the musical dimensions describing each successive note and interval of a subsequence can vary along the subsequence. Redundant structural information is filtered out without loss of information through closed and cyclic pattern mining [2].

In order to reduce the computation time, in the MIREX version, only two musical dimensions are considered:

- chromatic dimension: MIDI pitch of each note and inter-pitch interval between successive notes
- diatonic dimension: staff height of each note and corresponding interval between successive notes

In particular, gross contour is not taken into account, and no rhythmical analysis is performed.

1.2 *mus.output*: post-filtering and result formatting

A post-processing routine selecting the most important patterns and formatting the results into a text file is coded in the M file +mus/output.m.

Only patterns longer than 11 nodes [2] are selected.

The paradigmatic sheaf bundling described in section 3.2 in [2] is implemented in the function `sheaf`, also part of the M file +mus/output.m.

2. ALGORITHMIC DETAILS

2.1 Incremental one-pass approach

The approach is incremental, progressively analysing the musical sequence through one single pass. The main loop of this incremental approach (Algorithm 1) consists in a chronological scanning of the musical sequence: the analysis is carried out for each successive note of the sequence successively. This loop is coded in the `process_notes` function in the M file related to the `mus.minr` operator (/+mus/minr.m).

Algorithm 1 One-pass scanning of musical sequence

```
for each successive note $n_i$ in the sequence do
    ProcessNote($n_i$)
end for
```

[1] Freely available at *http://code.google.com/p/miningsuite/*.

[2] Each pattern is a branch in the pattern tree. The first node of each branch corresponds to the root of the tree, which is not associated to any note description. Hence patterns longer than 11 nodes correspond to sequence longer than 10 notes.

2.2 Occurrences extension

As explained in sections 2.2 and 3.3 in [2], integrating a new note consists in checking:

- whether pattern occurrence(s) ending at the previous note can be extended with the new note,
- whether the new note initiates the start of a new pattern occurrence.

These test are detailed in Algorithm 2. The two tests described above correspond respectively to the second loop (lines 6-14) and to the last operation (lines 15-16).

Each note is considered with respect to the previous note (since pitch intervals are integrated in the note description). For that reason, in the actual code, each note instantiates a syntagmatic connection between that note and its previous note, which is an object of the `pat.syntagm` class. The algorithm 2 is included in the constructor method of `pat.syntagm`.

Algorithm 2 ProcessNote(n_i)

```
for each cycle $C$ ending at $n_{i-1}$ do
  $P$ is the pattern related to $C$, $P_M$ is the periodic model and $\phi$ is the current phase on that period.
  ExtendCycle($C$, $n_i$,$P$,$P_M$,$\phi$)
end for
for each pattern occurrence $O$ ending at $n_{i-1}$, in decreasing order of specificity do
  $P$ is the pattern of $O$
  for each pattern $P_j$ equal to or more general than $P$, in decreasing order of specificity, except pattern equal to or more general than more general occurrences do
    ExtendOccurrence($O$, $n_i$, $P_j$)
  end for
end for
ExtendOccurrence([], $n_i$,$\emptyset$), where $\emptyset$ represents the pattern tree root, i.e. the empty pattern.
```

For each occurrence, the list of pattern extension operations is detailed in Algorithm 3. This algorithm is coded in the method `memorize` of the pattern occurrence class `pat.occurrence`.

2.3 Pattern Cyclicity

As explained in section 4 in [2], the successive repetitions of a periodic pattern leads to the formation of a single chain of states, where each state C is related to:

- a particular note n_i in the piece of music
- the *periodic model* P_M, which is the pattern representing the whole preceding period
- the pattern P that is constructed for that particular note. In the simple case, it is a prefix of the periodic model P_M at phase ϕ, i.e., $P = P_M(\phi)$
- the phase ϕ on that period: a phase $\phi = 0$ means that the periodic pattern is now completely reconstructed, so that a new occurrence of that pattern will start to be constructed from that point, a phase $\phi > 0$ indicates that the current note is associated with the ϕth note of the periodic pattern

Algorithm 3 ExtendOccurrence(O,n_i,P)

```
$d$ is note $n_i$'s description
for each possible extension $E$ of $P$ do
  if $E$'s description entirely conforms to $d$ then
    Extend $O$ with note $n$ following extension $E$.
  else if $E$'s description partially conforms to $d$ then
    $d'$ is the description common to $d$ and $E$
    if there is no extension of $P$ with description $d'$ then
      Create a new extension $E'$ of $P$ based on description $d'$.
      Extend $O$ with note $n$ following extension $E'$.
    end if
  end if
end for
Look at description $d$ in the continuation memory related to pattern $P$.
if there exists a similar older context (occurrence $O'$ followed by note $n'$) and $P$ extended with $d$ is closed then
  Create a new extension $E$ of pattern $P$ based on description $d$.
  Extend $O'$ with note $n'$ following extension $E$, and store the following note's description in the continuation memory related to $E$.
  Extend $O$ with note $n$ following extension $E$.
else
  Add the new note $n$ in the continuation memory.
end if
```

Algorithm 4 shows how to detect the successive phases and how to generalise progressively the period if needed. If at a given phase ϕ, the next note does not extend the pattern $P = P_M(\phi)$ following the description given by $P_M(\phi+1)$, but only partially, then for the next phase $\phi+1$, the new pattern P is more general than $P_M(\phi + 1)$. Once the whole period is generated, the resulting more general period P becomes the new periodic pattern P_M used as guide for the next period. This algorithm is coded in the method `track_cycle` in `pat.occurrence` class.

3. REFERENCES

[1] T. Collins. MIREX 2013: Discovery of Repeated Themes and Sections, 2013. *http://www.music-ir.org/mirex/wiki/2013:Discovery_of_Repeated_Themes_&_Sections* Accessed on 14 August 2014.

[2] O. Lartillot: "In-Depth Motivic Analysis Based on Multiparametric Closed Pattern and Cyclic Sequence Mining," *Proceedings of the International Symposium on Music Information Retrieval*, 2014.

Algorithm 4 ExtendCycle(C, n_i, P,P_M,ϕ)

```
d is note $n_i$'s description
for each possible extension $E$ of $P$ do
  $N \leftarrow []$
  if both $E$'s and $P_M(\phi+1)$'s descriptions entirely conform to $d$ then
    $N \leftarrow E$.
  else if both $E$'s and $P_M(\phi+1)$'s descriptions partially conform to $d$ then
    $d'$ is the description common to $d$ and $E$
    if there is no extension of $P$ with description $d'$ then
      Create a new extension $E'$ of $P$ based on $d'$.
    end if
    $N \leftarrow E'$
  end if
  if $N \neq []$ then
    if $\phi$ =length($P_M$) then
      $\phi' \leftarrow 0$, $P_M \leftarrow P$
    else
      $\phi' \leftarrow \phi + 1$
    end if
    Extend $C$: $C'$ with note $n_i$, pattern $N$, phase $\phi'$, pattern model $P_M$
  end if
end for
```