The Pendular Graph: Visualising Hierarchical Repetitive Structure in Point-set Representations of the POP909 Music Dataset

Chenyu Gao[1][0000−0002−8946−1668] and Tom Collins[2,3][0000−0001−7880−5093]

[1] University of York, York, UK
[2] University of Miami, Coral Gables, USA
[3] MAIA, Inc., Davis, USA
{chenyugao.cs,tomthecollins}@gmail.com

Abstract. Structure in music can mean many things: repetition, tonality, the existence of and focus on different "musical dimensions", such as rhythm, timbre, etc. Here, we are concerned with repetitive structures in music, such as sections that repeat within a song (verses, choruses, etc.). We are also concerned mainly with hierarchical repetition (e.g., within a verse, there may be a phrase or riff that recurs multiple times). Existing annotated music datasets tend to be either small in terms of items in the corpus, but with detailed annotatations, or larger as a corpus, but with linear annotations only. In this paper, we 1) develop a method for taking a linear annotation as input, and converting it to a hierarchical annotation as output, where such hierarchies exist in the input, and 2) introduce a web-based interface[4] where hierarchical annotations of 909 songs can be explored and played back, in synchrony with a visual representation of note content.

Keywords: First keyword · Second keyword · Another keyword.

1 Introduction

The word "structure" has multiple meanings in music theory, from repetition to the existence of and focus on different "musical dimensions" (e.g., pitch or rhythm). In this paper, we are mainly focusing on hierarchical repetitive structures in music. For example, there are sections (verses, choruses, etc.) that repeat within a song, in which there may be a phrase or riff that recurs multiple times within a verse. If a piece of music is annotated as "intro, verse, verse, chorus, verse, chorus, chorus" (or, more succinctly, "ABBCBCC"), this is a *linear* annotation (not hierarchical), because it contains no indication of smaller-scale repetition of phrases within a verse, and the existence of a larger section is only implicit (e.g., the substring "BC" occurs twice).

Comprehending musical structure has been shown to help people remember or understand music better [13], and could be especially useful for people listening to a piece for the first time. Visualised structure also provides a way to

[4] https://pendular-graph.glitch.me/

communicate intepretations of music, as different people may annotate the same song with different structural labels [3]. A lot of music structure visualisation tools have been developed [30,19,15,5,18,16,23], but many of these tools suffer either from a lack of interactivity, or are now unusable due to software dependency or maintenance issues. In this paper, we develop a Web-based hierarchical repetitive pattern visualisation tool based on the PatternViewer [23]. In our interface, the hierarchical structure of each song is visualised as a pendular graph, and repetitive phrases can be played by clicking nodes on the graph. Figure 1 shows an outline of our interface.

Existing music structure visualisation tools are highly relied on annotated data, while existing music datasets with structural annotations tend to be either a) small in terms of items in the corpus, but with detailed annotatations [6,26], or b) larger as a corpus, but with linear annotations only [11]. So, we define hierarchical structural annotations from an existing set of 909 linear-annotated popular Chinese songs.

The contributions of this paper are to 1) develop a method for taking a linear annotation as input, and hierarchicalise it to provide a hierarchical annotation as output (where such hierarchies exist in the input, such as with "BC" in the example above), and 2) introduce a web-based interface where hierarchical annotations of 909 songs can be explored and played back, in sync with the note-level content.

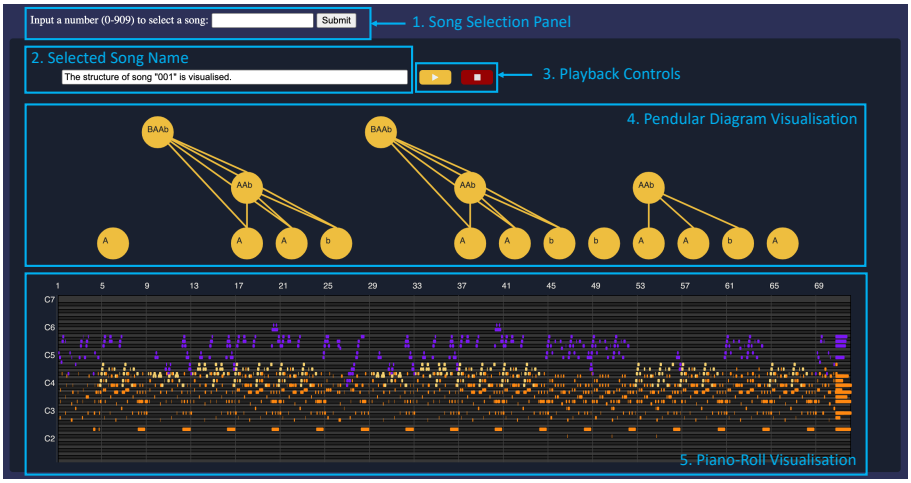

Fig. 1: A high-fidelity prototype of our interface.

2 Literature Review

This review is split into three parts: first, we provide a brief survey of relevant music representations and file formats; second, we consider previous work on preparing and publishing music datasets with structural annotations; third, we look to methods and tools developed for the visualisation of music-repetitive structure.

2.1 Relevant music representations and file formats

Mainly, this paper will overlook **audio** representations of music. Automatic transcription methods are improving [2,17] and offer the prospect of deriving note-level details from arbitrary audio inputs, but in this paper we are concerned with a dataset whose representation is symbolic (at the note level). Much work on symbolic music processing in the music information retrieval literature begins with melodic – or at least sequential – representations, such as ABC [27], viewpoints [8], or pitch/interval strings [9]. While some of the examples we include below from other researchers' work constitutes melodic or monophonic (meaning one note at a time) representations, generally, we are concerned with music where multiple notes can begin/end at the same time (polyphonic).

Musical Instrument Digital Interface (MIDI) is means via which certain electronic instruments or devices can communicate musical information. MIDI files contain the trace or record of people playing on such instruments, but they can also be created by someone inputting notes with a mouse or keyboard, using software such as a digital audio workstation. Note events belong to tracks that are associated with certain instruments, with the MIDI note number describing the pitch of a note, and "Note_on" and "Note_off" pairs indicating the start times and durations of notes. As such, MIDI does support polyphonic representation of music. Although other information, such as tempo and velocity (loudness), are also contained in MIDI, MIDI provides slightly different information to that contained in staff notation or sheet music (e.g., direction of note stems, voicing, pitch spelling, etc., are absent from MIDI files but present in staff notation).

MusicXML and MEI are hierarchical markup languages for representing staff notation, in much the same way that HTML represents text documents. They can also represent polyphonic music. Another difference between MusicXML and MIDI is that MusicXML describes musical events relative to one another whereas MIDI events are encoded in absolute terms. E.g., to calculate the start time of a note in a MusicXML file, it is necessary to know or calculate the start and end times of all preceding notes, whereas in a MIDI file it is not.

Composition, Instrument, and Production objects are three JSON-based file formats used to bridge audio and symbolic representations of music.[5] As implied by their names: a Composition object contains similar information to a digital score encoding or MIDI file; an Instrument object contains information about

[5] `https://musicintelligence.co/api/maia-spec/`

sample-based or synthesis-based sound generators; and a Production object contains information about audio effects that might be applied to instruments over the time course of a piece. For the purposes of this paper, we can focus on an example Composition object, given in Listing 1.

We will not go through the example in Listing 1 in full detail, but only mention the most relevant components. Lines 2-8 can be considered metadata, while lines 9-31 are the main musical data of the Composition object. The `layer` property straddles the categories of metadata and data, containing staff information as well as annotations of the piece of music. For instance, the `hierGt` property is an abbreviation of "hierarchical ground truth", and will be explained further in Section 3, since it is a new contribution of this paper. Each note in the piece is specified by an element of the `notes` array (lines 11-24), with properties that encompass both MIDI and MusicXML formats, depending on the source from which the Composition object is created. For example, if a Composition object is created via MusicXML import, a property such as `pitch` will be defined unambiguously; but if it is created via MIDI import, then `pitch` would have to be estimated from the context. On the other hand, if a Composition object is created via MIDI import, then there may be many tempo and/or control changes (such as sustain pedal), which can be stored in `tempi` and `controlChange` properties, respectively; but if it is created via MusicXML import, then the `controlChange` property is not likely to be populated.

2.2 Music Datasets with Structural Annotations

A classical music dataset named JKU-PDD [6] has repetitive theme and section (generally, "pattern") annotations. Ground-truth patterns in the JKU-PDD dataset are based on sectional repetitions marked in the score, as well as and music analysts' annotations [1,24,4]. A strength of the JKU-PDD is that it contains **hierarchical** patterns (e.g., an annotated theme may occur within an annotated section); a weakness is that only five pieces of music are labelled. Repetitive pattern annotations of another six pieces of classical music are provided by Tomašević et al. [26]. These six pieces are labelled by multiple annotators when studying to what extent different annotation tools and the background of annotators influence the annotation process. Similarly, the weakness of this dataset is that it only contains a handful of pieces.

The POP909 dataset [28] contains piano arrangements of 909 popular Chinese songs in MIDI format, with phrase-level repetitive structure labels [11]. So, in contrast to the JKU-PDD and Tomašević et al.'s annotations, a strength of POP909 is its size; but a weakness is that the annotations are not hierarchical – the labels are attributed at the phrase level only.

In the repetitive phrase labels, melodic phrases are labelled with capital letters, while non-melodic phrases are labelled with lower-case letters. A non-melodic introduction phrase is labelled with "i", a non-melodic ending phrase is labelled with "o", and other non-repetitive phrases are labelled with "X" and "x". The phrase length (the number of bars) is indicated by the number following each phrase label. For example, the song "123 Pinocchio" by Hey Girl in the

Listing 1 Simplified example of a Composition object for "123 Pinocchio".

```
{
  "id": "002",
  "name": "123 Pinocchio",
  "composers": [...],
  "layer": [
    ...,
    "hierGt": {...}
  ],
  "keySignatures": [...],
  "timeSignatures": [...],
  "notes": [
    {
      "barOn": 1,
      "beatOn": 1.5,
      "ontime": 0.5,
      "duration": 0.33333,
      "mnn": 75,
      "mpn": 68,
      "pitch": "D#5",
      "staffNo": 1,
      ...
    },
    ...
  ],
  "tempi": [...],
  "miscImport": {
    "midi": {
      "controlChange": [...]
    },
    ...
  }
}
```

POP909 dataset is labelled "i4A4A4B9b4A4B9b4B9X5o2", where "A4" means phrase "A" consists of 4 bars. Each time "A4" appears in the sequence, there is an exact or inexact repetition of the phrase "A" in the song.

In contrast to the JKU-PDD, with POP909, we do not know whether all or just some of the notes occurring in the four bars labelled "A4" belong to (are a repetition of) "A". But the size of POP909 and the potential for turning the annotations into a hierarchical repetitive structure (see Section 3) make it an interesting dataset.

2.3 Music Structure Visualisation Methods

The Shape of Song [29] is an application of arc diagrams [30], in which the repetition of a contiguous pitch sequence is connected to the previous occurrence with an arc. Figure 2 shows an example of the arc diagram, where the largest arc connects bars 1-2 with bars 5 and the first three pitches of bar 6. Two more D's in bars 3 and 7 could be considered included in the largest repetition, since the patterns are still perceptually similar after adding these two notes. However, the arc diagram limits repetitive patterns to be consecutive. If applied to polyphonic data, it would be necessary to define a sequential ordering of notes before the arc diagram could be calculated and constructed – and again, there could be issues with perceptually similar material not being annotated because of slight differences between occurrences of patterns. Similarly, the Infinite Jukebox [19] also connects repetitive patterns with arcs, but a song is visualised as a circle. The interactivity of the Shape of Song and the Infinite Jukebox is limited, and the arc diagrams become dense and difficult to interpret when the length and complexity of the music grows.

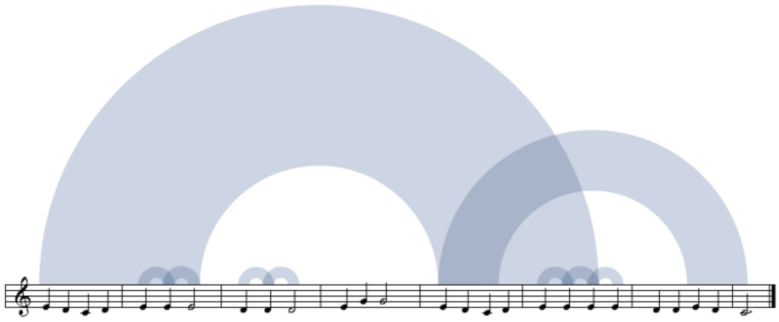

Fig. 2: "Mary Had a Little Lamb" visualised by using the Arc Diagrams [30].

Endrjukaite and Kosugi [15] visualises both repetitive patterns and changes in volume by using coloured cylindrical diagrams. Each song is represented by such a diagram, where repetitive patterns are drawn in the same colour, and the diameter reflects the volume. Similar to [15], Chen and Su [5] and Hayashi et al.

[18] also represent repetitive patterns in the same colours. Further, [5] introduces a clustering method to visualise the role of these repetitive patterns in the context of the whole piece of music, while [18] focuses on repetitions between different MIDI tracks. These visualisations suffer from limited interactivity.

By contrast, Songle [16] is more interactive, visualising both the repetitive structure of note and chord content of songs. Songle uses horizontal panels to represent structure and content. For example, one panel contains an automatic transcription of the note content, another an automatic transcription of the chords, while a third panel uses differently coloured oblongs to indicate repeated segments, with oblongs at the same vertical position and of the same colour being perceptually similar. But repetitive patterns are not organised by hierarchical structure, which makes it difficult for users to comprehend the overall structure of songs. Furthermore, mistakes in the automatic transcription can obscure relationships between song segments (e.g., a repeat chord sequence Em, G/D, C, B, D mistranscribed on occasion as Em, C7, Cdim7, G/D will not be identified as sufficiently similar to be highlighted visually).

PatternViewer [23] visualises the repetitive music structure in a pendular graph, and notes currently playing are coloured according to the estimated key. In a pendular graph, G, each vertex (or node) v in the set of vertices V represents a pattern occurrence $P_{i,j}$, where $P_{i,j}$ is a set of points belonging to a point-set representation of the entire piece. An occurrence set is denoted $\mathcal{P}_i = \{P_{i,j} \mid j = 1, 2, \ldots, m\}$, and in a visualisation of G, members of the same occurrence set $\mathcal{P}_i$ are shown as nodes with the same vertical location, with horizontal location determined by their order of appearance in the piece. Members of the same occurrence set are perceptually similar to one another – sometimes (but not necessarily) transnationally equivalent. Patterns containing more points are visualised vertically higher up, while patterns containing fewer points are lower down. An edge e connects two vertices u, v if the pattern occurrence $P_{i,j}$ associated with v is a subset of the pattern occurrence $P_{k,l}$ associated with u.

In PatternViewer, users can click nodes in the pendular graph to listen to a corresponding part of music. Unfortunately, only a handful of pieces were prepared to be visualised (due to the time-intensive nature of defining precisely which notes in a time window do/not belong to a pattern occurrence), and the current version of macOS does not support the installation of PatternViewer, since the software is no longer maintained.

3 Hierarchicalisation of the POP909 Annotations

Repetitive phrase labels for the POP909 dataset are proposed by Dai et al. [11]. The labels are only linear, however, while (in)exact repetition in music tends to be full of hierarchy. For instance, Figure 3 summaries the repetitive patterns of "123 Pinocchio". The largest repetitive pattern 'A4B9b4' appears twice, highlighted by the red bounding boxes. The labelling along the bottom part of Figure 3 is the same as that discussed in Section 2.2. The nodes and edges above this constitute an example of a pendular graph (see description

of the PatternViewer at the end of Section 2.3). For the sake of clarity, when displaying the graphs, we dispense with the bar numbering aspect (i.e., "A4" becomes "A").

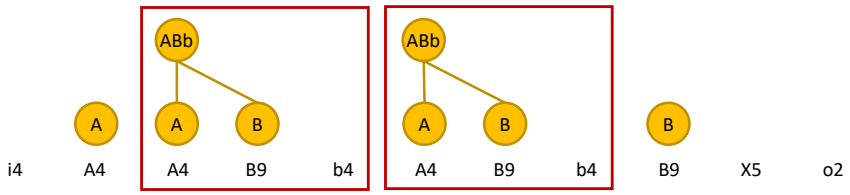

Fig. 3: Hierarchical structure analysis of "123 Pinocchio", Hey Girl. The smallest repetitive phrase is labelled by a letter followed by the number of bars. Phrases labelled with the same letter are (in)exact repetitive, and larger repetitive patterns are highlighted by bounding boxes. The nodes and edges constitue a pendular graph.

Algorithm 1 Hierarchicalises linear repetitive pattern labels.

Input: Linear, phrase-level repetitive pattern labels ($\mathbf{P}$) of the POP909 dataset
Output: Hierarchical repetitive pattern labels

for $p \in \mathbf{P}$ **do**
  Separate phrase labels and bar counts into two arrays as $\mathbf{V}_{\text{Phr}}$ and $\mathbf{V}_{\text{Bar}}$.
  $\mathbf{V}_{\text{Rep}} \leftarrow$ `get_repeat_subsequences(`p`)`
  for $v \in \mathbf{V}_{\text{Rep}}$ **do**
    if Non-repetitive patterns (e.g., "X", "x", "i", and "o") involved in v **then**
      Filter out v
    else if v only appears in larger patterns **then**
      Filter out v
    else
      if An occurrence of v overlaps the previous occurrence **then**
        Filter out the current occurrence of v
      end if
      if The occurrence count of $v > 1$ **then**
        Push v into $\mathbf{V}_{\text{Out}}$
      end if
    end if
  end for
  return $\mathbf{V}_{\text{Out}}$
end for

In this paper, inspired by Deutsch and Feroe [14], we propose an algorithm to convert linear, phrase-level repetitive pattern labels into a hierarchical labelling. Repetitive substrings with the longest length will be labelled as a new pattern.

E.g., phrases "ABb" in "123 Pinocchio" will be gathered as a group. On the other hand, substrings that never repeat (such as "i", "X" and "o") or substrings that appear only in a larger pattern will not be labelled. For example, "b" in "123 Pinocchio" will not be labelled since "b" never occurrs independently of "ABb". The outline of our algorithm is given in Algorithm 1. This algorithm takes linear pattern labels ($\mathbf{P}$) of songs as an input, and outputs hierarchical repetitive pattern labels.

When processing a sequence of letters and numbers p from $\mathbf{P}$, our algorithm begins by separating phrase labels (letters) and bar counts (numbers) into two arrays $\mathbf{V}_{\mathrm{Phr}}$ and $\mathbf{V}_{\mathrm{Bar}}$. The function `get_repeat_subsequences()` will return substrings of $\mathbf{V}_{\mathrm{Phr}}$ that repeat at least once, in descending order of length, as well as the index arrays where the first letter of these substrings occur in the entire string as $\mathbf{V}_{\mathrm{Rep}}$. For example, when inputting "abcabda", the function `get_repeat_subsequences()` outputs repetitive substrings "ab", and "a", and "b" and the index arrays of occurrences $[0, 3]$, $[0, 3, 6]$, and $[1, 4]$.

The following steps filter out repetitive patterns $v \in \mathbf{V}_{\mathrm{Rep}}$ that involve non-repetitive phrases (lines 5 and 6 of Algorithm 1), or appear only in larger patterns (lines 7 and 8). For example, "b" in the input "abcabda" will be filtered out, as "b" appears only in a larger pattern "ab". Occurrences of v will be filtered out if they overlap previous occurrences. Then, the number of occurrences of a repetitive pattern will be checked again after the filtering is complete, and only patterns whose occurrence count is larger than one will be pushed into the array $\mathbf{V}_{\mathrm{Out}}$. Finally, $\mathbf{V}_{\mathrm{Out}}$ will be returned for the piece of music. An example hierarchical structure annotation processed by our algorithm is shown in Listing 2, which is stored in the `layer` property of a Composition object (see end of Section 2.1).

The contents of lines 2 and 3 of Listing 2 should be compared with the bottom part of Figure 3: it is the linear annotation; meanwhile, the contents of lines 4-23 contain the pendular graph rendered in the top part of Figure 3.

4 Interface

Our interactive interface visualises the repetitive patterns of songs hierarchically, as a graph consisting of vertices and edges. The web interface is developed by using JavaScript packages: the client side is built mainly using p5.js; the server side is developed with Fastify and the Node Fetch APIs. The MAIA Util package [7] is used when processing music data, and Tone.js [21] enables the songs to be played back dynamically in the browser.

Figure 4 shows an architecture diagram, which is an overview of the work process of our interface. The whole process starts with a user inputting a number into the Song Selection Panel. The user's input will be sent to the server, and the Change Piece block will check if the input is an integer between 1-909. If the user's input is a sensible number, then a corresponding Composition Object will be requested from the Database, and the Composition Object as well as a message indicating the song is selected successfully will be sent back to the Song Selection Panel on the client side. Otherwise, a message with the reason

Listing 2 A hierarchical structure annotation example of "123 Pinocchio".

```
"hierGt": {
  "pLabel": "iAABbABbBXo",
  "bars": [4, 4, 4, 9, 4, 4, 9, 4, 9, 5, 2],
  "pendularGraph": {
    "ABb": {
      "edges": [
        "A",
        "B"
      ],
      "subsetScore": 0,
      "occ": [2, 5]
    },
    "A": {
      "edges": [],
      "subsetScore": 1,
      "occ": [1, 2, 5]
    },
    "B": {
      "edges": [],
      "subsetScore": 1,
      "occ": [3, 6, 8]
    }
  }
}
```

for the failure selection will be sent to the client side. After getting the message back from the server side, the Song Selection Panel will phrase the message, and decide whether to refresh the Pendular Diagram visualisation.

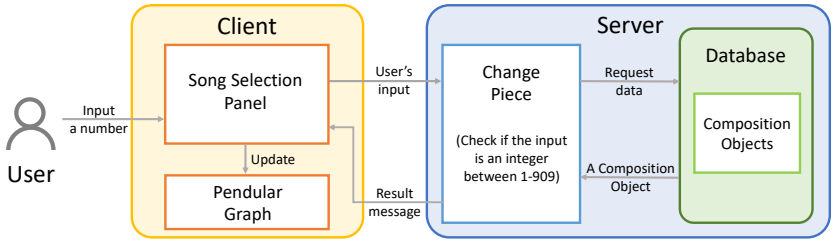

Fig. 4: An architecture diagram.

A high-fidelity prototype of our proposed interface is shown in Figure 1, which includes 5 elements:

1. **Song selection panel** at the top of the page. A user could enter a number between 1 and 909 in the text input box to select a song to visualise. Once the user clicks the 'Submit' button, the server side will read and process data of the selected song, and send it to the client side. Then, content related to the song structure visualisation will be refreshed.
2. **Selected song name** panel below the song selection panel, which tells the user which song is selected and visualised.
3. **Playback controls** can be clicked to start/pause and stop a song.
4. **Pendular graph visualisation** is the main component of our interface, so-called because the vertices/nodes resemble pendulums. The code behind this component parses the hierarchical structure labels annotated by our algorithm (Section 3), and draws them as a pendular graph, where each vertex represents a repetitive pattern. Horizontal location is determined by appearance in the piece; patterns containing more points are visualised vertically higher up, while patterns containing fewer points are lower down. An edge will connect two vertices if one pattern's point-set representation is a subset of that of another. Users can click nodes in the graph to listen to the corresponding part of the music.
5. **Piano-roll visualisation** at the bottom of this page, in which the melody is coloured in yellow, the secondary melody/lead instruments is coloured in purple, and the accompaniment is coloured in orange.

5 Discussion

When composing music, artists are carefully introducing repetition to emphasise music ideas [25]. Recognising repetitive patterns is also an important step

for understanding music [22,20]. Visualising the structure of music can enhance audiences' understanding of music, especially when they are listening to a piece of music for the first time [12]. A web-based interactive interface is developed by this paper, in which hierarchical repetitive structure of 909 pop songs can be visualised, and repetitive patterns can be played back by clicking nodes on the pendular graph. Compared with existing music structure visualisation tools, our interface has better interactivity, which can also visualise more songs than most other existing music structure visualisation tools.

During the data pre-processing process, we noticed that most of symbolic data (MIDI in particular) needs to be quantised carefully before using. For example, the downbeats of MIDIs in the POP909 dataset are not aligned to bar lines correctly, and time signature is missing. So, we first align the downbeats according to Dai's annotations [11]. To calculate the time signature of each song in the POP909 dataset, we assume the time signature of each song is either in 3/4 or 4/4. We could get how many bars each song has by using the phrase-level annotations as references, and the total count of crotchet beats can be known from the corresponding MIDI file. Our program estimates the time signature of each song by determining if the count of crotchet beats calculated by 'bars $\times$ beats per bar' matched that in the MIDI file. After the calculation, we found that the time signature of over 95% of songs in the POP909 dataset is 4/4, while only 28 songs are in 3/4. Quantisation for some symbolic data from the internet could be more complicated especially when MIDIs are recorded by using the MIDI keyboard directly. Thus, a general quantisation method for symbolic data is expected to be developed.

The lack of structure is a common problem in music generated by deep learning based models [10]. In this paper, we developed an interactive interface for visualising the hierarchical structure of the POP909 dataset, which is a quite popular dataset for music generation model training. We hope our interface could inspire researchers in music generation model design on enhancing the structure of generated music.

5.1 Limitations and Future Work

At the expense of relatively coarse levels of definition – i.e., pattern occurrences must begin/end at bar beginnings/endings, and they are defined temporally, meaning all notes existing between two time points are assumed members of a pattern occurrence (which is less precise than with the PatternViewer visualisations).

Another limitation of our work is that visualisation is restricted by the labelled datasets. The current version of the interface does not support users input an arbitrary song to visualise its structure. So, future work is to incorporate pattern discovery algorithms into this system, which will enable the users to the visualised structure of songs uploaded by themselves. Also, the incorporation of pattern discovery algorithms is hopefully to solve the problem of pattern occurrences must begin/end at bar beginnings/endings.

As we mentioned in previous sections, existing datasets with structural annotations are either small or large but with linear-level labels only. The lack of data for evaluation is a reason that restricts the development of pattern discovery algorithms. The POP909 annotations introduced in this paper can act as a ground truth to evaluate pattern discovery algorithms.

References

1. Barlow, H., Morgenstern, S.: A dictionary of musical themes. Crown Publishers (1948)
2. Benetos, E., Dixon, S., Duan, Z., Ewert, S.: Automatic music transcription: An overview. IEEE Signal Processing Magazine **36**(1), 20–30 (2018)
3. Bent, I.D., Pople, A.: Analysis. In: Sadie, S., Tyrrell, J. (eds.) The new Grove dictionary of music and musicians, vol. 1, pp. 526–589. Macmillan, London, UK, 2nd edn. (2001)
4. Bruhn, S.: JS Bach's Well-tempered Clavier: In-depth analysis and interpretation, vol. 4. Siglind Bruhn (1993)
5. Chen, T.P., Su, L.: The musical schemagram: Time-scale visualization of repeated patterns in music. In: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC). pp. 1642–1648. IEEE (2018)
6. Collins, T.: MIREX 2014 competition: Discovery of repeated themes and sections, 2014 (2015)
7. Collins, T., Coulon, C.: MAIA Util: an NPM package for bridging web audio with music-theoretic concepts. In: Proceedings of the web audio conference. pp. 47–52 (2019)
8. Conklin, D., Witten, I.H.: Multiple viewpoint systems for music prediction. Journal of New Music Research **24**(1), 51–73 (1995)
9. Crawford, T., Badkobeh, G., Lewis, D.: Searching page-images of early music scanned with omr: a scalable solution using minimal absent words (2018)
10. Dai, S., Yu, H., Dannenberg, R.B.: What is missing in deep music generation? a study of repetition and structure in popular music. In: Proceedings of 23rd International Conference on Music Information Retrieval (2022)
11. Dai, S., Zhang, H., Dannenberg, R.B.: Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music. In: Proceedings of the 2020 Joint Conference on AI Music Creativity (2020)
12. De Prisco, R., Malandrino, D., Pirozzi, D., Zaccagnino, G., Zaccagnino, R.: Understanding the structure of musical compositions: Is visualization an effective approach? Information Visualization **16**(2), 139–152 (2017)
13. Deutsch, D.: The processing of structured and unstructured tonal sequences. Perception & psychophysics **28**(5), 381–389 (1980)
14. Deutsch, D., Feroe, J.: The internal representation of pitch sequences in tonal music. Psychological review **88**(6), 503 (1981)
15. Endrjukaite, T., Kosugi, N.: Music visualization technique of repetitive structure representation to support intuitive estimation of music affinity and lightness. Journal of Mobile Multimedia pp. 049–071 (2012)
16. Goto, M., Ogata, J., Yoshii, K., Fujihara, H., Mauch, M., Nakano, T.: Podcastle and songle: Crowdsourcing-based web services for spoken document retrieval and active music listening. In: 2012 Information Theory and Applications Workshop. pp. 298–299. IEEE (2012)

17. Hawthorne, C., Simon, I., Swavely, R., Manilow, E., Engel, J.: Sequence-to-sequence piano transcription with transformers. In: Proceedings of 22nd International Society for Music Information Retrieval Conference (2021)
18. Hayashi, A., Itoh, T., Matsubara, M.: Colorscore: Visualization and condensation of structure of classical music. Knowledge Visualization Currents: From text to art to culture pp. 113–128 (2013)
19. Lamere, P.: The infinite jukebox. `https://eternalbox.dev/jukebox_index.html`
20. Lerdahl, F., Jackendoff, R.S.: A Generative Theory of Tonal Music, reissue, with a new preface. MIT press (1996)
21. Mann, Y.: Interactive music with tone. js. In: Proceedings of the 1st annual Web Audio Conference. Citeseer (2015)
22. Meyer, L.B.: Emotion and meaning in music. University of chicago Press (2008)
23. Nikrang, A., Collins, T., Widmer, G.: Patternviewer: An application for exploring repetitive and tonal structure. In: J.-SR Jang, M. Goto & JH Lee (Chairs), Late-Brake Demo of the Proceedings of the 15th International Society for Music Information Retrieval Conference. ISMIR, Taipei, Taiwan (2014)
24. Schoenberg, A.: Fundamentals of Musical Composition. Faber and Faber (1967)
25. Schoenberg, A.: Style and idea: Selected writings. Univ of California Press (2010)
26. Tomašević, D., Wells, S., Ren, I.Y., Volk, A., Pesek, M.: Exploring annotations for musical pattern discovery gathered with digital annotation tools. Journal of Mathematics and Music **15**(2), 194–207 (2021)
27. Walshaw, C.: The abc music standard 2.1. technical report. `https://abcnotation.com` (2011)
28. Wang, Z., Chen, K., Jiang, J., Zhang, Y., Xu, M., Dai, S., Bin, G., Xia, G.: Pop909: A pop-song dataset for music arrangement generation. In: Proceedings of 21st International Conference on Music Information Retrieval (2020)
29. Wattenberg, M.: Shape of song. `http://www.turbulence.org/Works/song/`
30. Wattenberg, M.: Arc diagrams: Visualizing structure in strings. In: IEEE Symposium on Information Visualization, 2002. INFOVIS 2002. pp. 110–116. IEEE (2002)