

Automatic Detection of Hierarchical Structure and Influence of Structure on Melody, Harmony and Rhythm in Popular Music

Shuqi Dai¹, Huan Zhang², and Roger B. Dannenberg³

Computer Science Department, Carnegie Mellon University
{shuqid, huanz, rbd}@cs.cmu.edu

A Appendix: Detailed Structure Analysis Algorithm

Our structure analysis begins with the identification of repetition of harmony, melody and rhythm. We then use a combination of dynamic programming, A* search, and heuristics to find good structure descriptions guided by the Structure Description Length (*SDL*) rating.

A.1 Data Pre-processing

We use a Chinese Pop song MIDI data set consisting of 909 transcriptions of audio, which is also analyzed (Wang et al., 2020). We use key and chord labels from audio in combination with labels automatically derived from MIDI. To improve the labeling, we manually resolved conflicts between automatic key labels in 135 of the songs. We also compared audio to MIDI chord labels after quantizing the chords into 2-beat units. To resolve conflicts, we rate the labels using $functionality(c, key) + similarity(c, prev) + similarity(c, next)$, where *functionality* (from *music21* (Cuthbert & Ariza, 2010)) rates the relevance of chord *c* to the key, and *similarity* (see in the main paper) is based on the number of common tones with the previous and next chords. Our MIDI files have a melody track, simplifying melody extraction. We quantized the melodies to 16ths. Our source data is not precisely aligned to beats, so we manually labeled and eliminated downbeat offsets so that every bar line aligns to the correct downbeat. Thus, we have accurate transcriptions and harmonic analyses of 909 Chinese pop songs.

A.2 Similarity Metrics

We use three different similarity metrics to detect repetitions in the song.

The *chord progression similarity metric* is based on the number of common pitch classes between chords. For example, given chord sequences "Cmaj Dmin" and "Cmin Ddim Dmin", Cmaj ({C, E, G}) and Cmin ({C, E $\flat$, G}) share 2 out of 3 notes, Dmin and Ddim also share 2 out of 3, and we consider the similarity of the extra Dmin to be 1, so the total score is $similarity_{chord} = \frac{2}{3} + \frac{2}{3} + 1 = 2.33$.

The *rhythm similarity metric* is based on matching note onset times. Onsets times are quantized to 16^{th} notes. Given two melodic sequences we define $similarity_{rhythm}$ to be the proportion of 16^{th} note offsets where both sequences or neither sequence contains an onset (i.e. *accuracy*). In our data, the MIDI notes are not 100% clean and aligned, and there are cases where two repeated segments have a 16^{th} note offset. So in practice, the rhythm similarity is defined as the the maximum accuracy over sequence offsets of +1, 0, and -1.

The *melody pitch contour similarity* is based on a Dynamic Time Warping (DTW) (Berndt & Clifford, 1994) algorithm applied to melody pitch contour. First, we transform two melody note sequences by sampling at a rate of 16^{th} notes. E.g. a note with pitch p and duration 4 in the original melody will become 4 consecutive copies of p in the new sequence. We compute the edit distance using a substitution cost based on similar absolute pitch difference and similar melodic direction. We normalize the DTW distance and subtract from 1 to get a similarity.

$$cost_{sub}(p_1, p_2) = w_1 \cdot dist_{pit}(p_1, p_2) + w_2 \cdot dist_{dir}(prev_1 - p_1, prev_2 - p_2), \quad (1)$$

$$\text{where } w_1 = 1.0 \text{ and } w_2 = 2.0 \text{ are weights,} \quad (2)$$

$$\text{and } prev_i \text{ is the pitch before } p_i. \quad (3)$$

$$dist_{pit}(p_1, p_2) = \begin{cases} 12, & p_1 \text{ or } p_2 \text{ is a rest,} \\ |p_1 - p_2|, & \text{otherwise.} \end{cases} \quad (4)$$

$$dist_{dir}(\Delta_1, \Delta_2) = \begin{cases} 0, & \Delta_1 = \Delta_2 \text{ or } \Delta_1 \cdot \Delta_2 > 0 \\ 2, & \Delta_1 \cdot \Delta_2 < 0 \\ & \text{or } \Delta_1 \text{ or } \Delta_2 \text{ is a rest} \\ 1, & \text{otherwise.} \end{cases} \quad (5)$$

The insertion cost for adding a pitch or rest p_X in sequence X compared to sequence Y is defined as:

$$cost_{ins}(p_X, Y) = \begin{cases} |p_X - \overline{p_Y}|, & p_X \text{ is a pitch} \\ 10, & p_X \text{ is a rest,} \end{cases}$$

where $\overline{p_Y}$ is the average pitch of Y .

The DTW calculation for the distance between sequences X_i and Y_i is

$$dist(X, Y) = dtw(|X|, |Y|), \text{ where}$$

$$dtw(i, j) = \min(dtw(i-1, j-1) + cost_{sub}(X_i, Y_j),$$

$$dtw(i-1, j) + cost_{ins}(X_i, Y),$$

$$dtw(i, j-1) + cost_{ins}(Y_i, X)).$$

Finally, to get from the DTW distance to a similarity, we normalize the DTW distance by the distances to pitch averages and subtract from one, being careful to make the lowest similarity 0:

$$\begin{aligned} \text{similarity}_{\text{melody}}(X, Y) &= \max\left(1 - \frac{2c}{a+b}, 0\right), \text{ where} \\ a &= \text{dist}(X, \bar{Y}), b = \text{dist}(Y, \bar{X}), \text{ and } c = \text{dist}(X, Y), \\ \text{where } \bar{X} &\text{ is the sequence } X \text{ with each pitch replaced} \\ &\text{by the average of pitches in } X. \end{aligned}$$

Finding Repetition To determine if a segment of music repeats an earlier segment, we use thresholds:

$$\text{match}(A, B) = \text{similarity}_{\text{melody}}(A, B) \geq 0.6 \wedge \text{similarity}_{\text{chord}} \geq 0.65 \wedge \text{similarity}_{\text{rhythm}} \geq 0.9$$

A.3 Algorithm Description

Given a song consisting of melody, a chord analysis, and a time signature, determine the song structure with the best structure description length (*SDL*). The algorithm follows:

Step 1. Find all matched phrase pairs (repetitions). We used the *match* function above to compare every pair of song segments of length 4 to 20 measures. All segment pairs considered must not overlap and must have the same phrase length at this step.

Step 2. Merge matching pairs into sets of matching phrases. If we view each of the phrases in our matched phrase pairs as a node in an undirected graph and add edges between the phrases that are matched, then finding all the sets is equivalent to finding all maximal cliques in this sparse undirected graph. We call these *phrase sets*.

Step 3. Find best structure using dynamic programming and A* search. If we treat each matching phrase set as a node in an undirected graph, and add an edge between two nodes if they can be used in a structure of the song together, then the problem is equivalent to the maximum weighted clique problem in the undirected graph. Since this is an NP-complete problem, we combine dynamic programming, A* search, and heuristics to create a good solution with reasonable efficiency.

Input: phrase sets $M = \{m_1, m_2, \dots, m_{|M|}\}$ of song S , sorted in ascending order of the first measure of the first phrase in the set. Each set m_i is represented by a set of (typically disjoint) measures. Our implementation uses bit vectors to represent phrase sets.

If a structure is a set of *phrase sets*, define a *substructure* to be a *subset* of phrase sets, i.e. segments including repetitions that might only partially cover all the measures of the song. We define a substructure's *cover state* $\mathbb{C}(\Omega) = c_1 c_2 c_3 \dots c_l$, where l = number of measures in song S , $c_i = 1$ when the i^{th}

measure is an element of some phrase set in Ω , and otherwise $c_i = 0$. Multiple substructures can have the same cover state. For each cover state C , the optimal structure description length using the first i phrase sets in M (and covering remaining measures with Xn segments) is $f(C, i)$. For convenience, we write $SDL(C)$ to mean $SDL(\Omega)$ for some Ω where $\mathbb{C}(\Omega) = C$. The dynamic programming inference formulation is:

$$\begin{aligned} f(C \cup m_i, i) = \min(&f(C \cup m_i, i - 1), \\ &f(C, i - 1) - \Delta SDL(C, m_i)), \end{aligned} \quad (6)$$

where m_i does not overlap C , i.e. $m_i \cap C = \emptyset$, and

$$\Delta SDL(C, m_i) = SDL(C) - SDL(C \cup m_i)$$

Note that $\Delta SDL(C, m_i)$ can be computed efficiently by considering the added cost due to m_i . Algorithmically, the computation maintains an array of objects, each with a unique cover state and its minimum known SDL : $[C, x]$, where $x = SDL(C)$. An outer loop takes each m_i in order, finds cover states that do not overlap, and constructs new states $[C \cup m_i, SDL(C \cup m_i)]$. These new states replace an existing state if the new SDL is lower, and new states are added if there is no existing state with the same cover set. (See Algorithm 1.)

Algorithm 1: Dynamic Programming

Result: Set of states and their lowest SDLs

$states \leftarrow \emptyset$;

for $m_i \in M$ **do**

for $[C, x] \in states$ **do**

if $C \cap m_i = \emptyset$ **then**

 update $states$ with $[C \cup m_i, SDL(C \cup m_i)]$;

end

end

end

A* Pruning. As the search progresses, the minimum SDL over all the states establishes an upper bound:

$$UB = \min_{[C, sdl] \in states} (sdl) \quad (7)$$

on the SDL of the optimal structure. Although we can possibly improve the SDL of known substructures by covering more measures with additional phrase sets, there are limits to how much improvement is possible. If we can show that a state can never lead to an SDL below UB , we can simply prune the state.

More precisely, what is the largest ΔSDL obtainable by adding a phrase set to a state? Recall that M is sorted according to the left-most covered measure. At any point in the algorithm, remaining phrase sets can only cover measures from some left-most measure k to the last one. Let t be the number of these uncovered measures and d be a phrase length in our allowable range of 4 to 20 measures:

$$\begin{aligned}\Delta SDL_{ub} &\leq g(t-d) + h(1-t/d) \\ &\leq g(t-4) + h(1-t/4)\end{aligned}\tag{8}$$

If a state's $SDL - \Delta SDL_{ub} > UB$, then we can safely prune the state.

Step 4. Adjust structure boundaries. Notice that there might be a difference in length of one or two measures between matched phrases. In this step, we adjust phrase boundaries by looking at extra unmatched measures. If any measure can be combined with the previous phrase and stay within the similarity metrics threshold, we extend the phrase by one measure.

A.4 Complexity, Optimizations, and Evaluation

Let n be the number of measures of the song. In step 1, there are $O(n^2)$ matched pairs and since pairs have a maximum length of 20, the time for computing the similarity score for each pair is $O(1)$. The total time complexity for this step is $O(n^2)$. In Step 2, there can be at most $O(n^2)$ phrase sets. Although the maximal cliques problem is NP-complete, even long songs have only hundreds of measures and the number of phrase sets grows roughly linearly with song length. According to our experiment, the actual number of phrase sets $|M|$ is a linear function of n . To find all matching phrase sets, we iterate through all matching phrase pairs ($O(n^2)$) and update them to the existing matching sets. Notice that each matching sets can have at most $O(n/4)$ phrases, thus, the total time complexity for this step is $O(n^2 \cdot n/4 \cdot |M|) = O(n^4)$. For step 3, the time complexity is $O(|C|^2|M|) = O(2^{2n} \cdot n)$, but again due to the limited number of C that we encountered during the dynamic programming process ($|C|_{max} = 56256$ in the experiment), the run time is still durable. To further optimize the algorithm performance, we designed the following approaches:

1. *Exclude phrases* that begin with melody and end with more than two non-melody measures, forcing a split and reducing the number of matching phrases.
2. Transform all operations in the algorithm into *bit manipulation* operations.
3. Take the top 10-20% of the phrase sets with the largest covering and use the best SDL for the A* pruning step using all phrase sets.
4. *Approximation option 1.* In each decile of M (ordered by starting measure), pick 10-20% with the largest covering size. Run the algorithm on these phrase sets.
5. *Approximation option 2.* In each decile of M , pick the 10-20% with the highest similarity metrics scores. Run the algorithm on these phrase sets.
6. *Approximation option 3.* Randomly pick 10% of M and get the best possible result. Run the random process 10 to 20 times and pick the best result among them.

Our full algorithm correctly produced 93% of the human-labeled structures (Table 1). In the evaluation, we only consider the exact same structure representations as the correct match, there might be some under-segment or over-segment

cases that is partially right, but we all label them as incorrect analysis. The average run time of each song on a laptop with a 2.3GHz 8-Core Intel Core-i9 and a 64GB-2667MHz-DDR4 RAM is 345s, but for 80% of the songs, the average run time is only 21s. Approximation option 1 has an accuracy of 68% and a run time of only 13s.

Full Algorithm	Approx1	Approx2	Random
92.86%	67.86%	35.71%	39.29%

Table 1: Evaluation on Structure Analysis Experiments.

References

- Berndt, D., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Aaaiws'94: Proceedings of the 3rd international conference on knowledge discovery and data mining* (pp. 359–370).
- Cuthbert, M., & Ariza, C. (2010). Music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the international society for music information retrieval conference* (p. 637-642).
- Wang, Z., Chen, K., Jiang, J., Zhang, Y., Xu, M., Dai, S., . . . Xia, G. (2020). Pop909: A pop-song dataset for music arrangement generation..