Deriving Representative Structure Over Music Corpora

Ilana Shapiro, Ruanqianqian Huang, Zachary Novack, Cheng-I Wang,
Hao-Wen Dong, Taylor Berg-Kirkpatrick, Shlomo Dubnov, Sorin Lerner

▶ **To cite this version:**

Ilana Shapiro, Ruanqianqian Huang, Zachary Novack, Cheng-I Wang, Hao-Wen Dong, et al.. Deriving Representative Structure Over Music Corpora. 2024. hal-04722377

HAL Id: hal-04722377
https://hal.science/hal-04722377v1

Preprint submitted on 5 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Synthesizing Composite Hierarchical Structure from Polyphonic Music Corpora

Ilana Shapiro[1], Ruanqianqian (Lisa) Huang[1], Zachary Novack [1], Cheng-i Wang [1], Hao-Wen Dong [1], Taylor Berg-Kirkpatrick [1], Shlomo Dubnov [1], Sorin Lerner [1]

[1]University of California, San Diego, CA, USA
{ilshapiro, r6huang, znovack, chw160, h3dong, sdubnov, tberg, lerner}@ucsd.edu

Abstract

Western music is an innately hierarchical system of interacting levels of structure, from fine-grained melody to high-level form. In order to analyze music compositions holistically and at multiple granularities, we propose a unified, hierarchical meta-representation of musical structure called the *structural temporal graph* (STG). For a single piece, the STG is a data structure that defines a hierarchy of progressively finer structural musical features and the temporal relationships between them. We use the STG to enable a novel approach for deriving a representative structural summary of a music corpus, which we formalize as a dually NP-hard combinatorial optimization problem. Our approach first applies simulated annealing to develop a measure of *structural distance* between two music pieces rooted in graph isomorphism. Our approach then combines the formal guarantees of SMT solvers with nested simulated annealing over structural distances to produce a structurally sound, representative *centroid* STG for an entire corpus of STGs obtained from individual pieces. To evaluate our approach, we conduct experiments showing that structural distance accurately differentiates between music pieces, and that our computed centroids encapsulate the overarching structure of a music corpus.

1 Introduction

A prevailing theory among Western music theorists and musicologists states that Western classical music exhibits an implicitly hierarchical structure (Simonetta et al. 2018). While several different theoretical systems have been proposed to formalize this structural hierarchy (Marsden, Hirata, and Tojo 2013), a widely accepted modern interpretation of the hierarchy states that melodies form the bottom, followed by harmonic contour, rhythmic patterns, disjoint and possibly overlapping motifs, and finally large-scale sections (Nieto 2015; Mount 2020; Dai, Zhang, and Dannenberg 2024). Together, this composite hierarchy encapsulates the overall structure of a piece.

To analyze musical structure computationally, many automatic approaches have been developed for extracting structure at *single* levels of the structural hierarchy(McFee et al. 2017; Hsiao et al. 2023; Levé et al. 2011; Chen and Su 2021; Salamon et al. 2014). However, music perception researchers have shown that the levels are not perceptually independent as they relate to one another in both "vertical" (structural) and "horizontal" (temporal) directions (Narmour 1983), interactions that have more recently been proven computationally (Dai, Zhang, and Dannenberg 2024). A comprehensive analysis of a piece thus must integrate these inter- and intra-level interactions into a unified model.

Despite prior attempts at an integrated computational model of musical form, two challenges remain. First, prior approaches do not completely encapsulate the vertical and horizontal relationships of the structural hierarchy, cannot handle polyphonic music, or are not fully automatic (Hamanaka, Hirata, and Tojo 2016; Simonetta et al. 2018; Mokbel, Hasenfuss, and Hammer 2009; Carvalho and Bernardes 2020). Second, to our knowledge, existing methodologies focus only on *individual* pieces, and hence there has been no attempt to *summarize* the hierarchy over a music corpus to synthesize its overall structure and obtain a holistic music representation of the entirety of the corpus.

To address these challenges, we introduce the *structural temporal graph* (STG) as a unified model of polyphonic musical structure. The STG is a k-partite directed acyclic graph whose levels form the structural hierarchy, and edges convey temporal relationships between adjacent levels. We then use simulated annealing to develop a measure of *structural distance* between two STGs based on graph isomorphism. Finally, to obtain the overarching structure of a corpus of pieces, we develop an approach for deriving a representative *centroid* graph from a corpus of STGs, which jointly minimizes structural distance and standard deviation over the corpus. We formalize the centroid derivation as a bi-level NP-hard combinatorial optimization problem, and propose a solution combining nested simulated annealing with the formal guarantees of SMT solvers in order to produce a structurally sound result. An evaluation of our approach shows that the structural distance measure can differentiate between pieces, and that centroids can effectively characterize the music corpora they are derived from.

In summary, the contributions of this paper are as follows:

1. We propose the *structural temporal graph*, an integrated meta-representation of musical form that unifies the music structural hierarchy.
2. We develop a *structural distance* measure between two STGs rooted in graph isomorphism.
3. We formalize the music summarization problem as a dually NP-hard combinatorial optimization problem, and

contribute a novel approach to solving this problem using both stochastic and SMT-based techniques.

4. We conduct experiments verifying that the structural distance can differentiate pieces, and that music corpora are accurately characterized by their derived centroids.

2 Related work

2.1 Single-Level Analyses

Many successful algorithms have been developed to extract structure at *single* levels of the music structural hierarchy. To extract segmentation, The Music Structure Analysis Framework (MSAF) toolkit (Nieto 2015) features factorization-based techniques, including ordinal linear discriminant analysis (McFee and Ellis 2014b), convex nonnegative matrix factorization (Nieto and Jehan 2013), checkerboard (Foote 2000), spectral clustering (McFee and Ellis 2014a), the Structural Features algorithm (Serrà et al. 2014), 2D-Fourier Magnitude Coefficients (Nieto and Bello 2014), and the Variable Markov Oracle (Wang and Mysore 2016).

Motif discovery algorithms search for disjoint, repeating, and possibly overlapping patterns in a piece. String-based approaches (Wang, Hsu, and Dubnov 2015) represent music as a chromagram and detect patterns with sub-string matching, and geometry-based approaches (Hsiao et al. 2023) represent music as multidimensional point sets, and translatable subsets identify patterns. Recent approaches in harmony identification are centered around neural networks, such as using transformers to incorporate chord segmentation into the recognition process (Chen and Su 2019, 2021). Until very recently, the Melodia algorithm was the state of the art in melody extraction, but recent approaches have shifted to neural networks (Kosta et al. 2022; Chou et al. 2021).

2.2 Integrated Models of Structure

Music theorists have attempted to unify the structural hierarchy with frameworks such as Schenkerian theory (Marsden 2010) and the Generative Theory of Tonal Music (GTTM) (Lerdahl and Jackendoff 2020). Schenkerian analysis results from applying a series of reductions that progressively simplify a musical piece by removing layers of structure. Attempts to automatically derive Scherkerian analyses are intractable for all but very short pieces, and have low accuracy (Marsden 2010). GTTM generates four different structural hierarchies (grouping structure, metrical structure, time-span tree, and prolongational tree) for a piece of music, to model human cognition (Hamanaka, Hirata, and Tojo 2016). Computational implementations GTTM (e.g. the Automatic Timespan Tree Analyser (Hamanaka, Hirata, and Tojo 2016)) cannot handle polyphonic music, and are not fully automatic. Improved results using these theories are unlikely, as neither provides a degree of precision required for complete computational implementation (Marsden, Hirata, and Tojo 2013).

The limitations of these theories have led researchers to turn to a modern interpretation of the structural hierarchy: segmentation, motifs, rhythm, harmony, and melody (Nieto 2015; Mount 2020; Dai, Zhang, and Dannenberg 2024), where motifs refer to disjoint, repeating patterns. Several attempts have been made to capture this hierarchy in graph data structures, such as by using automatic topographic mapping to represent melodic progressions (Mokbel, Hasenfuss, and Hammer 2009), encoding interactions between sections, melody, harmony and rhythm in a graph (Dai, Zhang, and Dannenberg 2020), and by using an undirected graph to represent melodic segments and their reductions (Orio and Rodà 2009). In addition, the prototype graph (Young, Du, and Bastani 2022) encodes music structure as a bipartite network of relationships between prototype elements and the music they represent. Finally, attempts have been made to model the structure hierarchy with formal grammars (Sidorov, Jones, and Marshall 2014; Finkensiep et al. 2023), but they are limited to segmentation and motifs.

None of these approaches encapsulate the entire hierarchy, and to our knowledge, there have also been no attempts to synthesize a representative, summarizing structure from a music corpus.

3 Structural Temporal Graph

3.1 Overview

To address the lack of a fully automatic complete encapsulation of polyphonic musical structure, we introduce the *structural temporal graph* (STG), a unified meta-representation of musical structure that captures the levels of the music structural hierarchy and the temporal relationships between them. The STG is a k-partite directed acyclic graph (DAG), where each of the k layers encodes a level in the music structural hierarchy.[1] Following the modern music theoretic interpretation of the hierarchy (Nieto 2015; Mount 2020; Dai, Zhang, and Dannenberg 2024), from top to bottom we denote the levels to be contiguous segmentation, motifs (both disjoint and overlapping), rhythmic contour, relative keys, functional harmonic chords, and melodic contour.

The STGs we build include every level in the hierarchy except rhythmic contour, as we were unable to access an algorithm to generate this analysis. We run individual analysis algorithms to generate each level of the hierarchy, which is elucidated in Section 6.1. Before giving a formal definition of the STG, we build intuition by walking through an example of deriving an STG from an annotated piece.

3.2 Building the Graph

We walk through the derivation of an STG from Beethoven's Biamonti Sketch No. 461 that unifies contiguous segmentation; disjoint, overlapping motifs; relative keys; functional harmonic chords; and melodic contour. First, we manually analyze the piece by marking up its score with ground-truth hierarchical annotations in Figure 1. Each colored annotation corresponds to one level of the structural hierarchy. In purple, we see that this piece has one large contiguous segment (numbered as segment 0). Next, we mark disjoint motifs in red. Motif 0 appears twice, at the beginning of bars

[1] Individual levels themselves can form sub-hierarchies of increasing granularity, commonly seen in segmentation, which the STG supports.

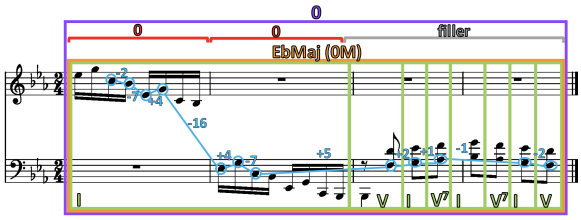

Figure 1: Ground-truth human analysis of Beethoven's Biamonti Sketch No. 461. The purple markings show the first level of the hierarchy (segmentation), the red markings show the second level (motifs) with the gray "filler" label indicating no motifs in that interval, the orange markings show relative key number and quality (M for major) in the third level, the green markings show the fourth level (functional harmony) using Roman numeral chord symbols, and the blue markings show the bottom level (melodic contour) by delineating the intervals between individual melody notes.

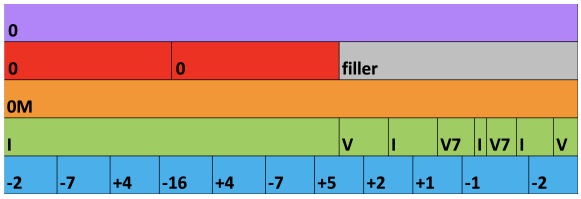

Figure 2: Spatial visualization of the analysis in Figure 1

1 and 2. The gray filler bar indicates no more motifs appear in the latter half. In orange, we see this piece is in a single key, Eb Major. We label this key symbolically as 0M to indicate this is relative key number 0 (i.e. the first key) in M for major. Subsequent keys would be numbered by their positive interval difference from the previous key within the 12-tone scale. We next see functional harmonic chords in green annotated with Roman numeral chord symbols, and finally melodic contour intervals in blue. Then, we equivalently represent the ground-truth annotations in Figure 1 as the stacked rectangles in Figure 2 to more clearly see how each level of the structural hierarchy relates to the next.

Keeping Figure 2 in mind, we make the transition from the human annotations in Figures 1 and 2 to the computer-generated STG for this piece in Figure 3. The edges and nodes of the STG, respectively, correspond to the vertical and horizontal alignments of the rectangles in Figure 2. We see that all motif nodes, including the gray filler node indicating no motifs for that interval, fall into the time interval of purple segmentation node 0. The orange key node 0M begins in the first red motif node 0, and ends in the last gray motif filler node (i.e. the key spans the entire piece). All the green chord nodes fall in the orange key node's interval. Finally, we see how blue melodic contour nodes relate to green chord nodes. For instance, the first melody interval -2 begins in the first chord node I, whereas the penultimate melody interval -1 begins in the fourth V7 chord node, and ends in the fifth I chord node.

Note that there are minor discrepancies between the green

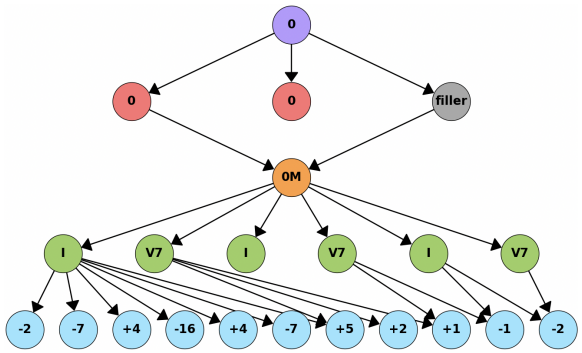

Figure 3: Computer-generated STG for Beethoven's Biamonti Sketch No. 461

functional harmony and blue melody nodes in Figure 3 and the rectangles in the same color in Figure 2. This is because Figure 2 shows a human-generated ground-truth analysis, while each level of the STG in Figure 3 is generated by a different music analysis algorithm. The STG is a fully automatic *meta*-representation of musical structure, and so any STG is only as accurate as the analysis algorithms it uses.

Formally, the nodes of an STG encode labeled musical sections along with their associated time intervals generated by the relevant analysis algorithm. Nodes are sorted within each level based on start time, and edges encode temporal relationships between nodes of adjacent levels. Specifically, for node n at level i, n must have either one or two parents in level $i-1$ directly above it: one if its associated time interval is a total subset of its parent's, and two if its time interval begins in one parent's, and ends in the other's.

4 Structural Distance

4.1 Graph Augmentation

At a high level, the distance between two STGs is is the minimum number of edit operations (deletion, insertion, and substitution of nodes and edges) required to transform one graph to the other, also known as the graph edit distance (GED) (Serratosa 2021). However, GED measures *isomorphic* similarity between two graphs, meaning it evaluates how closely the graph structures match independent of labeling. We cannot currently leverage STG isomorphism because STGs are "compressed," with structure encoded into node ids. Specifically, this compressed structure is found in the defining features of each node id, and in the intra-level linear temporal orderings for each analysis (i.e. the horizontal order of each level in the graph, currently determined by node index). Thus, in order to reason about STGs isomorphically, we must augment them to encode all structural attributes directly within the graph's topology.

To encode element labels, recall that each node id encodes a defining feature set. All nodes can thus be alternatively encoded as *instances* of their feature *prototypes*. We create a *prototype node* for each feature and assign it as a parent of the corresponding instance node(s) with that feature. For instance, segmentation nodes encode a single feature: the section number they correspond to. Finally, to

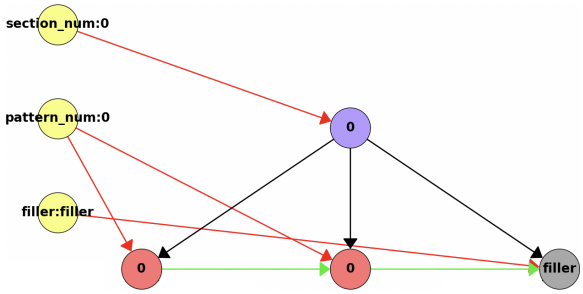

Figure 4: Augmenting the first two levels of the Beethoven Biamonti No. 461 STG. The yellow prototype nodes have the format `feature_name:feature_value`.

encode intra-level linear temporal relationships, we form a linear chain with edges between pairs of horizontally adjacent nodes. This results in a structurally complete STG we can reason about isomorphically. The first two levels of the STG from Figure 3 are shown in Figure 4, with yellow prototype nodes on the left for each instance feature (section number for segmentation, and pattern number and filler for motifs), red edges connecting prototype features to their instance nodes, and green edges for the intra-level linear chain in the motif layer.

4.2 Graph Alignment Annealing

GED is a NP-hard combinatorial optimization problem (Serratosa 2021), intractable for typical STGs. Most GED approximation algorithms are slow and of dubious accuracy, and more generalized than we require. We thus introduce our own difference measure called *structural distance* computed with simmulated annealing (SA). SA is a stochastic optimization technique used to estimate the global optimum of a discrete cost function. It comprises an objective "energy" function to minimize and a "move" function for generating a new solution from the current state. An annealer begins at a high *temperature* indicating the likelihood of accepting worse solutions to explore the solution space, and ends at a low temperature to refine near-optimal solutions. (Guilmeau, Chouzenoux, and Elvira 2021)

To use SA, we first convert the augmented STGs to adjacency matrices, and pad each matrix with zero-arity "dummy nodes" such that they have identical dimensions. Given such matrices A_{G_1} and A_{G_2}, their distance is given in Equation 1, where $\|.\|_F$ denotes the Frobenius norm.

$$\textsc{Dist}(A_{G_1}, A_{G_2}) = \|A_{G_1} - A_{G_2}\|_F \tag{1}$$

Finding permutation matrix P optimally aligning A_{G_2} to A_{G_1} to minimize Equation 1 is NP-hard, so we use SA. The alignment annealer's energy function is given in Equation 2. Given optimal P, Equation 2 computes the structural distance between A_{G_1} and A_{G_2}.

$$\textsc{Energy}(A_{G_1}, A_{G_2}, P) = \textsc{Dist}(A_{G_1}, P^T A_{G_2} P) \tag{2}$$

The move function for modifying P at each step of SA is given in Algorithm 1. A partition is either the set of instance nodes at a single level in the STG (e.g. the set of functional harmonic key nodes), or the set of prototype nodes for a given feature (e.g. chord quality).[2] Permuting only within valid partitions leverages the STG's inherent structure to avoid invalid moves globally detrimental to Equation 2.

Algorithm 1: Alignment Annealer Move Function

function MOVE
    Choose random index i in P
    Choose random index j from the same partition to which i belongs
    Swap rows i and j in P
end function

We set the alignment annealer's initial state to $P = I$, the identity matrix. By running the annealer for sufficiently many steps, we obtain optimal P.

5 Centroid Derivation

5.1 Bi-Level Centroid Annealing

We now derive a representative "centroid" STG from a corpus of STGs that minimizes both structural distance and variance over the corpus, such that the centroid is both maximally close and similar to the corpus. Specifically, given the padded adjacency matrix A_g for a centroid STG and its associated corpus of matrices $C = \{A_G\}$ that are optimally aligned to A_g, we want to minimize LOSS in Equation 3c, where DIST is as in Equation 1.

$$\mu_{\text{dist}}(A_g, C) = \frac{1}{|C|} \sum_{A_G \in C} \textsc{Dist}(A_g, A_G) \tag{3a}$$

$$\sigma_{\text{dist}}(A_g, C) = \sigma(\{\textsc{Dist}(A_g, A_G) \mid A_G \in C\}) \tag{3b}$$

$$\textsc{Loss}(A_g, C) = (\mu_{\text{dist}}(A_g, C) \cdot (\sigma_{\text{dist}}(A_g, C) + \epsilon))^2 \tag{3c}$$

Mean and standard deviation both factor into the loss as we seek for the centroid to be both maximally close and similar to the corpus. We square their result so the annealer is more sensitive to large changes the loss than smaller ones. Finally, we offset σ_{dist} by ϵ so the loss is still sensitive to changes in μ_{dist} when σ_{dist} is zero.

Finding the centroid A_g minimizing Equation 3c is again a NP-hard constraint satisfaction problem requiring SA. The centroid annealer's energy function is given in Equation 4

$$\textsc{Energy}(A_{G_1}, A_{G_2}, P) = \textsc{Loss}(A_g, C_{\text{aligned}}) \tag{4}$$

where C_{aligned} is the corpus aligned to the current centroid A_g, a process itself requiring SA as in Section 4.2 to obtain the optimal alignments. Centroid annealing is thus a dually NP-hard problem (GED and constraint satisfaction over these distances) requiring nested SA.

Finally, note that as the centroid annealer's temperature cools, the loss converges as the centroid becomes increasingly closely aligned to the corpus. Thus, as the *centroid*

[2] It is possible for an optimal alignment to match prototype nodes across different features in an instance level's feature set, i.e. matching a prototype in A_{G_1} with a prototype in A_{G_2}, but this is highly unlikely for larger graphs, and smaller individual feature partitions significantly benefit performance.

1. No self-loops
2. No instance-prototype or prototype-prototype edges
3. No edges from a prototype to an instance whose feature set does not include the proto feature (e.g. melody interval sign proto-segmentation instance)
4. No edges from lower to higher level instance levels (must respect the hierarchy)
5. No edges between non-adjacent instance levels (must respect k-partite structure)

Table 1: Global Constraints

annealer's temperature cools, we can scale down the number steps and maximum temperature of the nested *alignment* annealer at each iteration of the centroid annealing.

The centroid annealer's move function for modifying the centroid A_g at each step of SA is given in Algorithm 2. To

Algorithm 2: Centroid Annealer Move Function

function MOVE
  Calculate the list of absolute difference matrices $D_L = [|A_g - A_{G_i}| \text{ for } A_{G_i} \in C_{aligned}]$
  Calculate the element-wise mean difference matrix M and standard deviation matrix S over D_L
  Calculate the score matrix $X = M \cdot (S - \epsilon)$. Higher score at coord (i, j) means that coord has a higher impact on the loss
  Flatten X and sort in descending score order
  Partition X_{flat} by unique score, and shuffle each partition randomly (increases variability of moves)
  Iterate through the indices (i, j) of the sorted, partition-shuffled X_{flat}. Stop at the first (highest score) (i, j) such that flipping the (i, j) edge in A_g is not:
    - a globally structurally invalid move
    - a move undoing the most recently accepted move (avoid oscillating states)
    - a move the annealer has already locally rejected since the last accept (avoid getting stuck)
  Execute move: $A_g[i, j] = 1 - A_g[i, j]$
end function

move strategically, we build the score matrix S revealing which edge(s) in A_g contribute most to the loss. We add or remove the edge at a highest score coordinate meeting the criterion in Algorithm 2. In particular, a globally structurally invalid move induces a terminally invalid structure in the centroid by violating one of the rules in Table 1. Some locally invalid moves, however, such as removing an edge in an intra-level linear chain, must be allowed as intermediate steps to a more optimal structurally valid state. Importantly, the STGs being compared must have the same number of levels; otherwise, edges spanning multiple levels must be allowed as they can be intermediate states towards the deletion of an entire level. Based on our experiments, this would be unacceptably detrimental to the performance of the annealer.

We set the centroid annealer's initial state A_g to the STG in the corpus with the minimum loss over the rest of the

1. Every instance node must have 1 or 2 instance parents in the level above
2. The instances nodes at level l must form a linear chain/total ordering via intra-level edges
3. The start and end nodes of the linear chain must have the previous level linear chain's start and end nodes, respectively, as parents.
4. In instance levels with non-overlapping nodes,[3] the *first* parent of a node at linear chain index $i > 0$, must not come before node $i-1$'s *last* parent in the previous instance level's linear chain
5. The *first* parent of an instance node at linear chain index $i > 0$, must not come before node $i - 1$'s *first* parent in the previous level's linear chain

Table 2: Instance Constraints

1. Every instance node must have exactly one prototype parent per feature
2. For levels that require it,[4] no two linearly adjacent instance nodes can have identical prototype parent sets

Table 3: Prototype Constraints

corpus. By running the annealer for sufficiently many steps, we derive an approximate centroid for the corpus that may contain locally invalid states.

5.2 Graph Repair with SMT Solving

In order to obtain a structurally sound centroid, we must "repair" the approximate centroid A_g by projecting it to the nearest valid STG. We achieve this by encoding the STG's structure as constraints in quantifier-free first-order logic formulae in the SMT (satisfiable modulo theory) solver Z3 (de Moura and Bjørner 2008), which gives us formal guarantees on the soundness of the centroid. We use Z3's optimizer to minimize an objective over the constraints. Given approximate centroid A_g and valid centroid A'_g, our objective is given in Equation 5.

$$\text{OBJ}(A_g, A'_g) = \sum_i \sum_j |A_{g_{ij}} - A'_{g_{ij}}| \tag{5}$$

Our constraints include the global rules in Table 1, as well as additional constraints for instance nodes in Table 2, and finally prototype nodes in Table 3. We track relationships between nodes with uninterpreted functions.

Z3's optimizer supports integration with large neighborhood search (LNS) and can return intermediate semi-optimized solutions after a timeout. We run the optimizer with LNS, with initial soft constraints set to the approximate centroid A_g to guide the optimizer. Even so, naively running the optimizer on a full STG is generally intractable due to combinatorial explosion, so we partition A_g into subsets we can apply the constraints to incrementally.

[3] Segmentation, keys, chords, and melody, but not motifs

[4] Segmentation, keys, and chords only

We first partition A_g into pairs of consecutive instance levels without their prototypes (e.g. a segmentation/motif pair of instance levels), and optimize the instance constraints in Table 2 alongside the relevant global constraints in Table 1 over each partition incrementally. We combine the result of one partition with the previous until we build a valid centroid subgraph of instance nodes. Then, we partition A_g into single levels, each containing the instance nodes of that level and all prototype nodes for each instance feature at that level (e.g. segmentation instance nodes + all section number prototype nodes). The instance constraints are already optimized, so we now only need to optimize the prototype constraints in Table 2 and the relevant global constraints in Table 1 over the possible prototypes. This gives us the complete, structurally sound centroid A'_g we seek.

A more detailed visualization of the centroid derivation process is found in Appendix A.

6 Evaluation

6.1 Experimental Setup

We build a corpus of polyphonic, symbolic MIDI piano music from the Kunstderfuge dataset and the Classical Piano MIDI Database, [5] which we select for their high-quality data. Since some single-level analyses we use to generate STGs require the data to be in audio and CSV format, we convert MIDI to CSV (with a manual script) and to MP3 (with Fluidsynth (Moebert 2007)). This gives us a triple of paired MIDI, MP3, and CSV for each piece.

Recall from Section 3.2 that our STGs encapsulate segmentation, motifs, relative keys, functional harmonic chords, and melodic contour. To build the segmentation layer, we use the flat Structural Features algorithm (Serrà et al. 2014) to determine segment boundaries and 2D-Fourier Magnitude Coefficients (Nieto and Bello 2014) to determine segment labels, both of which are provided by the Music Structure Analysis Framework (Nieto 2015). To extract motifs, we use the BPS-motif discovery algorithm (Hsiao et al. 2023), and for relative keys and functional harmonic chords we use the pretrained Harmony Transformer V2 (Chen and Su 2021). Finally, to generate melodic contour we use the Melodia algorithm (Salamon et al. 2014).

Our experiments validate two objectives: (1) that the structural distance accurately differentiates between individual pieces, and (2) that the centroid graph encapsulates the overarching structure of the corpus it is derived from.

6.2 Evaluation of Structural Distance

From our STG dataset, we construct 210 sets built from 32 pieces by the composers J.S. Bach, Mozart, Beethoven, Schubert, and Chopin (21 Bach, 2 Mozart, 3 Beethoven, 2 Schubert, and 4 Chopin). Each set contains a unique combination of 5 pieces, one from each composer, such that the duration of any piece is within 7 seconds of any other piece in the same set, in order to normalize for piece length. We then use our graph alignment annealer from Section 4 to compute the pairwise structural distances between the individual STGs derived from the pieces within each set, resulting in 210 structural distance matrices, and take their mean[6]. For each pairwise structural distance, we run the graph alignment annealer for 2000 iterations with a max and min temperature parameters of 2 and 0.01, respectively. Our infrastructure is a Linux system of 8 NVIDIA GeForce RTX 2080 Ti GPUs, each with 11GB RAM.

Metric	ρ_s	p-value
Structural Distance (ours)	**0.8207**	**0.0140**
SWAS	0.5775	0.1760
MIDI Features	0.4681	0.3150

Table 4: Mantel Test with Spearman's rank correlation coefficient for normalized mean distance matrices

We evaluate our approach against two baselines over the same 210 piece combinations. Baseline 1 is the mean distance matrix obtained by taking the cosine similarity between feature vectors extracted from each MIDI file using Music21 (Cuthbert 2006). The feature vectors encode up to 100 pitches, note durations, and note onset times; 1000 chords; and 10 sections per piece. Baseline 2 is the mean distance matrix over the pairwise Stent weighted audio similarities (SWAS) over the paired MP3 file for each piece[7]. SWAS is a composite audio similarity metric comprising zero-crossing rate, rhythm, chroma, spectral contrast, and perceptual similarity metrics, which we weight equally. Our ground-truth reference is the stylistic similarity indices between composers derived from the binomial index of dispersion calculated using human annotations and metadata from "The Classical Music Navigator" database (Smith and Georges 2014).

We normalize all matrices to a range between 0 and 1, and invert the similarity matrices to distances. Finally, we apply the Mantel test with Spearman's rank correlation coefficient ρ_s to evaluate our results against the human-annotated notions of music similarity from the Classical Music Navigator. We choose Spearman's metric over the more common Pearson's since the relationships between our data are best described by a monotonic ranking rather than a linear relationship. Our results are shown in Table 4. We attain both the highest ρ_s and lowest p-value for structural distance, verifying that the structural distance accurately differentiates between pieces and captures human conception of musical similarity.

6.3 Evaluation of Centroid

We generate centroid STGs for corpora containing pieces by Bach (6), Beethoven (9), Haydn (4), and Mozart (5), respectively. In order to introduce slightly more variance over each corpus, we relax the duration constraint from the previous experiment such that now the pieces in each corpus are

[5] Kunstderfuge: https://www.kunstderfuge.com/, Classical Piano MIDI Database: http://www.piano-midi.de/

[6] We use the simanneal Python package for SA: https://github.com/perrygeo/simanneal

[7] SWAS is computed with the AudioSimilarity package: https://github.com/markstent/audio-similarity

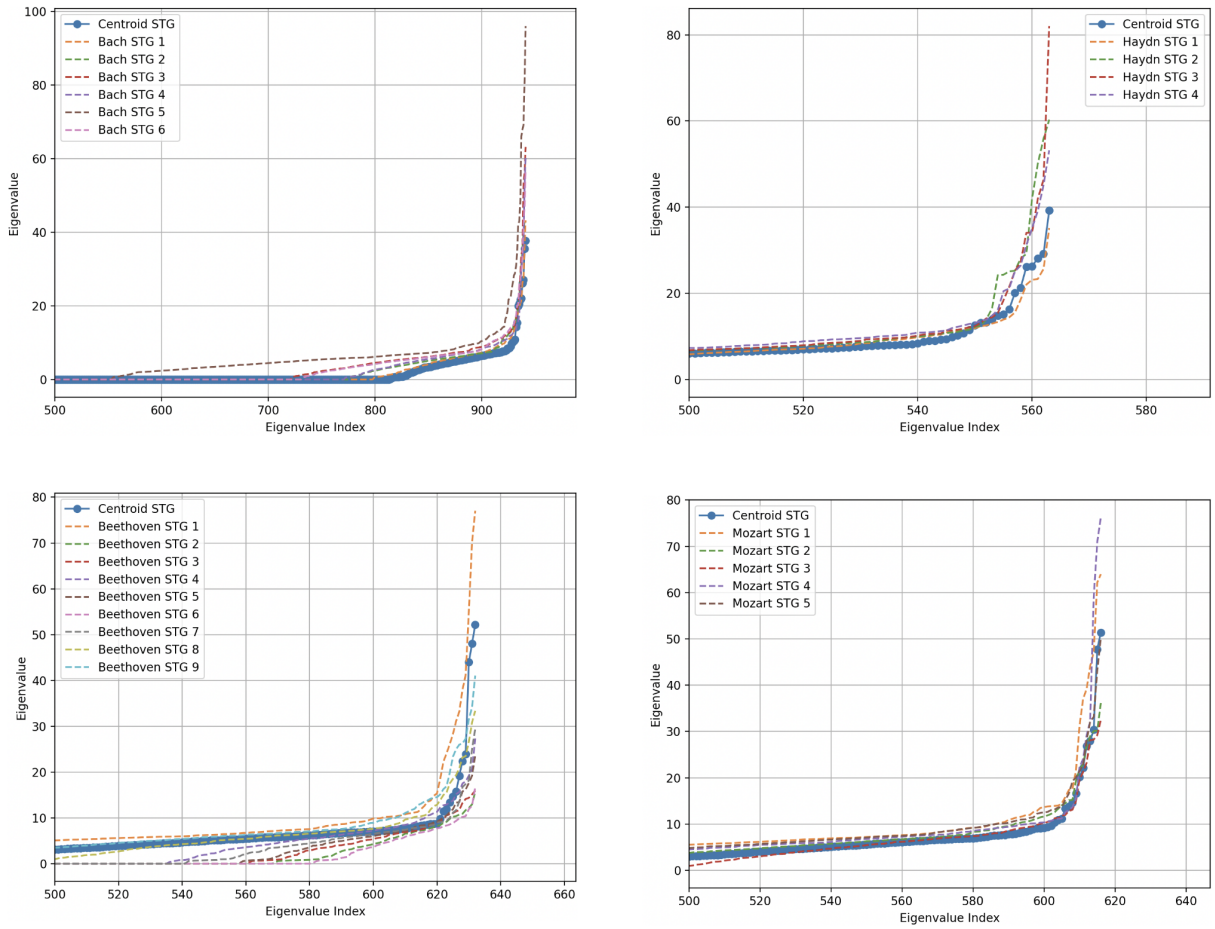

Figure 5: Spectra of the input STGs and computed centroid for each composer corpus

Composer	r
Bach	0.9732
Beethoven	0.9234
Haydn	0.9787
Mozart	0.9836

Table 5: Mean Pearson correlation r between the spectra of the centroid and corpus graphs for each composer

within 36 seconds of each other. We use the same GPU infrastructure as in the previous experiment to run the centroid annealer over each corpus and generate approximate centroids. Our GPUs struggled to handle generating centroids for more than 9 STGs due to memory limits, which impacted how we selected each corpus. Then, we run our Z3 optimizer to generate the final, structurally sound centroids on a Linux system with 24 Intel Xeon cores and 32GB RAM.

To evaluate whether our computed centroids effectively characterize their corpora, for each corpus we do a spectral decomposition of the adjacency matrix of each STG and the centroid. The spectrum of an adjacency matrix, or the set of its eigenvalues, encodes essential aspects of the graph's structure, making it a suitable tool for our evaluation. For each corpus, in Figure 5 we plot the spectra of each input STG and derived centroid together. For visibility, we truncate the graphs at lower eigenvalues, since they remain constant. We then calculate the mean Pearson's correlation coefficient r between the spectra of the centroid and each STG in the corpus, for all four composer corpora, in Table 5. We confirm a strong correlation for all composers, validating that the centroid effectively captures the essential structural characteristics of its corpus.

7 Conclusion and Future Work

We presented the *structural temporal graph* (STG), a unified meta-representation of the complete music structural hierarchy. The STG serves as a vital contribution to the holistic cognition and analysis of musical form. We used the STG to develop a measure of *structural distance* between two pieces rooted in graph isomorphism to reason about STGs quantitatively using simulated annealing. Finally, we framed the music stuctural summarization problem as a dually NP-hard combinatorial optimization problem, and contributed a novel approach combining nested simulated annealing with SMT solving to derive a structurally sound *centroid* STG

representative of a music corpus.

We envision applications of the STG and derived centroids in human-machine co-creation, in which a user refines a generated piece by modifying its STG to impose updated structural constraints on a generative model. In addition, one could use the centroid STGs to constrain a generative model to produce music with both local and global relational structure adhering to that of the chosen corpus. Finally, the STG and centroid derivation are not limited to music. The STG could easily model a structural hierarchy for any kind of hierarchical sequence data, so long as there exist algorithms to generate the analyses at each level. For instance, an STG could encapsulate the poetry structure hierarchy, with levels such as verses, stanzas, and lines, and the derived centroid would structurally summarize a poetry corpus. Together, these applications can inform opportunities in human modification of machine-generated data conforming to the desired structural specification.

8 Acknowledgments

This project is partially supported by the European Research Council under Europe's Horizon 2020 program, grant 883313 (ERC REACH) .

References

Carvalho, N.; and Bernardes, G. 2020. Towards balanced tunes: A review of symbolic music representations and their hierarchical modeling. In Cardoso, F. A.; Machado, P.; Veale, T.; and Cunha, J. M., eds., *Proceedings of the Eleventh International Conference on Computational Creativity, ICCC 2020, Coimbra, Portugal, September 7-11, 2020*, 236–242. Association for Computational Creativity (ACC).

Chen, T.; and Su, L. 2019. Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition. In Flexer, A.; Peeters, G.; Urbano, J.; and Volk, A., eds., *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, 259–267.

Chen, T.-P.; and Su, L. 2021. Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models.

Chou, Y.-H.; Chen, I.-C.; Chang, C.-J.; Ching, J.; and Yang, Y.-H. 2021. MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding. arXiv:2107.05223.

Cuthbert, M. 2006. music21.

Dai, S.; Zhang, H.; and Dannenberg, R. B. 2020. Automatic Analysis and Influence of Hierarchical Structure on Melody, Rhythm and Harmony in Popular Music. *CoRR*, abs/2010.07518.

Dai, S.; Zhang, H.; and Dannenberg, R. B. 2024. The Interconnections of Music Structure, Harmony, Melody, Rhythm, and Predictivity. *Music & Science*, 7: 20592043241234758.

de Moura, L. M.; and Bjørner, N. S. 2008. Z3: An Efficient SMT Solver. In Ramakrishnan, C. R.; and Rehof, J., eds., *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, 337–340. Springer.

Finkensiep, C.; Haeberle, M.; Eisenbrand, F.; Neuwirth, M.; and Rohrmeier, M. 2023. Repetition-Structure Inference With Formal Prototypes. In Sarti, A.; Antonacci, F.; Sandler, M.; Bestagini, P.; Dixon, S.; Liang, B.; Richard, G.; and Pauwels, J., eds., *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*, 383–390.

Foote, J. 2000. Automatic audio segmentation using a measure of audio novelty. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, volume 1, 452–455 vol.1.

Guilmeau, T.; Chouzenoux, E.; and Elvira, V. 2021. Simulated Annealing: a Review and a New Scheme. In *IEEE Statistical Signal Processing Workshop, SSP 2021, Rio de Janeiro, Brazil, July 11-14, 2021*, 101–105. IEEE.

Hamanaka, M.; Hirata, K.; and Tojo, S. 2016. *Implementing Methods for Analysing Music Based on Lerdahl and Jackendoff's Generative Theory of Tonal Music*, 221–249. Cham: Springer International Publishing. ISBN 978-3-319-25931-4.

Hsiao, Y.; Hung, T.; Chen, T.; and Su, L. 2023. BPS-Motif: A Dataset for Repeated Pattern Discovery of Polyphonic Symbolic Music. In Sarti, A.; Antonacci, F.; Sandler, M.; Bestagini, P.; Dixon, S.; Liang, B.; Richard, G.; and Pauwels, J., eds., *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*, 281–288.

Kosta, K.; Lu, W. T.; Medeot, G.; and Chanquion, P. 2022. A deep learning method for melody extraction from a polyphonic symbolic music representation. In Rao, P.; Murthy, H. A.; Srinivasamurthy, A.; Bittner, R. M.; Repetto, R. C.; Goto, M.; Serra, X.; and Miron, M., eds., *Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022*, 757–763.

Lerdahl, F.; and Jackendoff, R. 2020. *A Generative Theory of Tonal Music*. MIT Press.

Levé, F.; Groult, R.; Arnaud, G.; Séguin, C.; Gaymay, R.; and Giraud, M. 2011. Rhythm Extraction from Polyphony Symbolic Music. In Klapuri, A.; and Leider, C., eds., *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, 375–380. University of Miami.

Marsden, A. 2010. Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3): 269–289.

Marsden, A.; Hirata, K.; and Tojo, S. 2013. Towards computable procedures for deriving tree structures in music : context dependency in GTTM and Schenkerian theory.

McFee, B.; and Ellis, D. 2014a. Analyzing Song Structure with Spectral Clustering. In Wang, H.; Yang, Y.; and Lee, J. H., eds., *Proceedings of the 15th International Society*

for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014, 405–410.

McFee, B.; and Ellis, D. P. W. 2014b. Learning to segment songs with ordinal linear discriminant analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, Florence, Italy, May 4-9, 2014*, 5197–5201. IEEE.

McFee, B.; Nieto, O.; Farbood, M. M.; and Bello, J. P. 2017. Evaluating Hierarchical Structure in Music Annotations. *Frontiers in Psychology*, 8.

Moebert, T. 2007. FluidSynth: A SoundFont Synthesizer.

Mokbel, B.; Hasenfuss, A.; and Hammer, B. 2009. Graph-Based Representation of Symbolic Musical Data. volume 5534, 42–51. ISBN 978-3-642-02123-7.

Mount, A. 2020. *Fundamentals, Function, and Form*. Milne Open Textbooks.

Narmour, E. 1983. Some major theoretical problems concerning the concept of hierarchy in the analysis of tonal music. *Music Perception: An Interdisciplinary Journal*, 1(2): 129–199.

Nieto, O. 2015. *Discovering Structure in Music: Automatic Approaches and Perceptual Evaluations*. Ph.D. thesis, New York University.

Nieto, O.; and Bello, J. P. 2014. Music segment similarity using 2D-Fourier Magnitude Coefficients. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 664–668.

Nieto, O.; and Jehan, T. 2013. Convex non-negative matrix factorization for automatic music structure identification. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, 236–240. IEEE.

Orio, N.; and Rodà, A. 2009. A Measure of Melodic Similarity based on a Graph Representation of the Music Structure. In Hirata, K.; Tzanetakis, G.; and Yoshii, K., eds., *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, 543–548. International Society for Music Information Retrieval.

Salamon, J.; Gomez, E.; Ellis, D. P. W.; and Richard, G. 2014. Melody Extraction from Polyphonic Music Signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2): 118–134.

Serratosa, F. 2021. Redefining the Graph Edit Distance. *SN Comput. Sci.*, 2(6): 438.

Serrà, J.; Müller, M.; Grosche, P.; and Arcos, J. L. 2014. Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity. *IEEE Transactions on Multimedia*, 16(5): 1229–1240.

Sidorov, K. A.; Jones, A.; and Marshall, A. D. 2014. Music Analysis as a Smallest Grammar Problem. In Wang, H.; Yang, Y.; and Lee, J. H., eds., *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, 301–306.

Simonetta, F.; Carnovalini, F.; Orio, N.; and Rodà, A. 2018. Symbolic Music Similarity through a Graph-Based Representation. In Cunningham, S.; and Picking, R., eds., *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion, Wrexham, United Kingdom, September 12-14, 2018*, 26:1–26:7. ACM.

Smith, C. H.; and Georges, P. 2014. Composer similarities through "The classical music navigator": Similarity inference from composer influences. *Empirical Studies of the Arts*, 32(2): 205–229.

Wang, C.; Hsu, J.; and Dubnov, S. 2015. Music Pattern Discovery with Variable Markov Oracle: A Unified Approach to Symbolic and Audio Representations. In Müller, M.; and Wiering, F., eds., *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, 176–182.

Wang, C.-i.; and Mysore, G. J. 2016. Structural segmentation with the Variable Markov Oracle and boundary adjustment. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 291–295.

Young, H.; Du, M.; and Bastani, O. 2022. Neurosymbolic Deep Generative Models for Sequence Data with Relational Constraints. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 37254–37266. Curran Associates, Inc.

A Guided Example of Centroid Generation

In order to better comprehend the centroid derivation process in Sections 4 and 5, we walk through the five steps of an example centroid derivation from a corpus of two STGs in Figure 6, found at the bottom of this appendix. Each of the following steps corresponds to the analogously labeled step in Figure 6.

(i) We start with the corpus we will derive the centroid from. Our corpus contains two STGs. The first STG is the same as in Figure 3: Beethoven's Biamonti Sketch No. 461. The second STG is from Beethoven's Biamonti Sketch No. 811. Both STGs were built using the process in Section 3.

(ii) We augment both STGs from (i) as in Section 4.1. The implicit structure encoded in the node labels in (i) is now explicitly encoded in the graph's topology in (ii) in Figure 6 via the yellow prototype nodes, red prototype-instance edges, and green intra-level linear chains. The instance node labels thus are updated to only show the layer kind now (S for segmentation, P for patterns/motifs, K for keys, C for chords, or M for melody) Recall that each yellow prototype parent node encodes a *feature* of its instance child, and each prototype label is of the form `feature_name:feature_value`. The names of the features and their possible values for each level of the STGs in Figure 6 are explicated in Table 6.

(iii) Since all structure is now explicit and we can reason about the graphs isomorphically, we can now apply the nested simulated annealing procedure from Sections 4.2 and 5.1 to obtain the approximate centroid STG for the corpus. Some locally invalid states remain, such as the lack of the intra-level linear chain in the purple segmentation level.

(iv) We arrive at the final, well-formed centroid STG by projecting the approximate centroid STG from (iii) onto the nearest valid STG as in Section 5.2, using the SMT solver Z3 to impose guarantees of structural soundness on the result. Notice, for instance, how the requisite linear chain has been added to the purple segmentation level.

(v) Finally, we convert the repaired centroid from (iv) back to its compressed form (i.e. we move the information encoded in the prototype nodes and edges and the intra-level linear chains, back into the nodes labels), which allows us to visually compare it to the original STGs in the corpus from (i).

Level	**Feature Name**	**Feature Values**
Segmentation	`section_num`	$\mathbb{Z}_{\geq 0}$
Motifs	`pattern_num`	$\mathbb{Z}_{\geq 0}$
Motifs	`filler`	filler
Keys	`relative_key_num`	$\mathbb{Z}_{\geq 0}$
Keys	`quality`	{M, m}
Chords	`quality`	{M, m, d, d7, h7, D7, a, a6, a7}
Chords	`degree1`	$\{1, 2, \ldots, 12\}$
Chords	`degree2`	$\{1, 2, \ldots, 12\}$
Melody	`abs_interval`	$\mathbb{Z}$
Melody	`interval_sign`	$\{+, -\}$

Table 6: Prototype feature names (middle) and possible values (right) for each instance level (left). The chord quality values are shorthand for major, minor, augmented, diminished, half-diminished, dominant, etc. degree1 and degree2 refer to the primary and secondary degrees, respectively, in the diatonic scale that the chord is built on. Putting it all together, an example prototype label could be degree1:1, quality:M, filler:filler, or abs_interval:5.

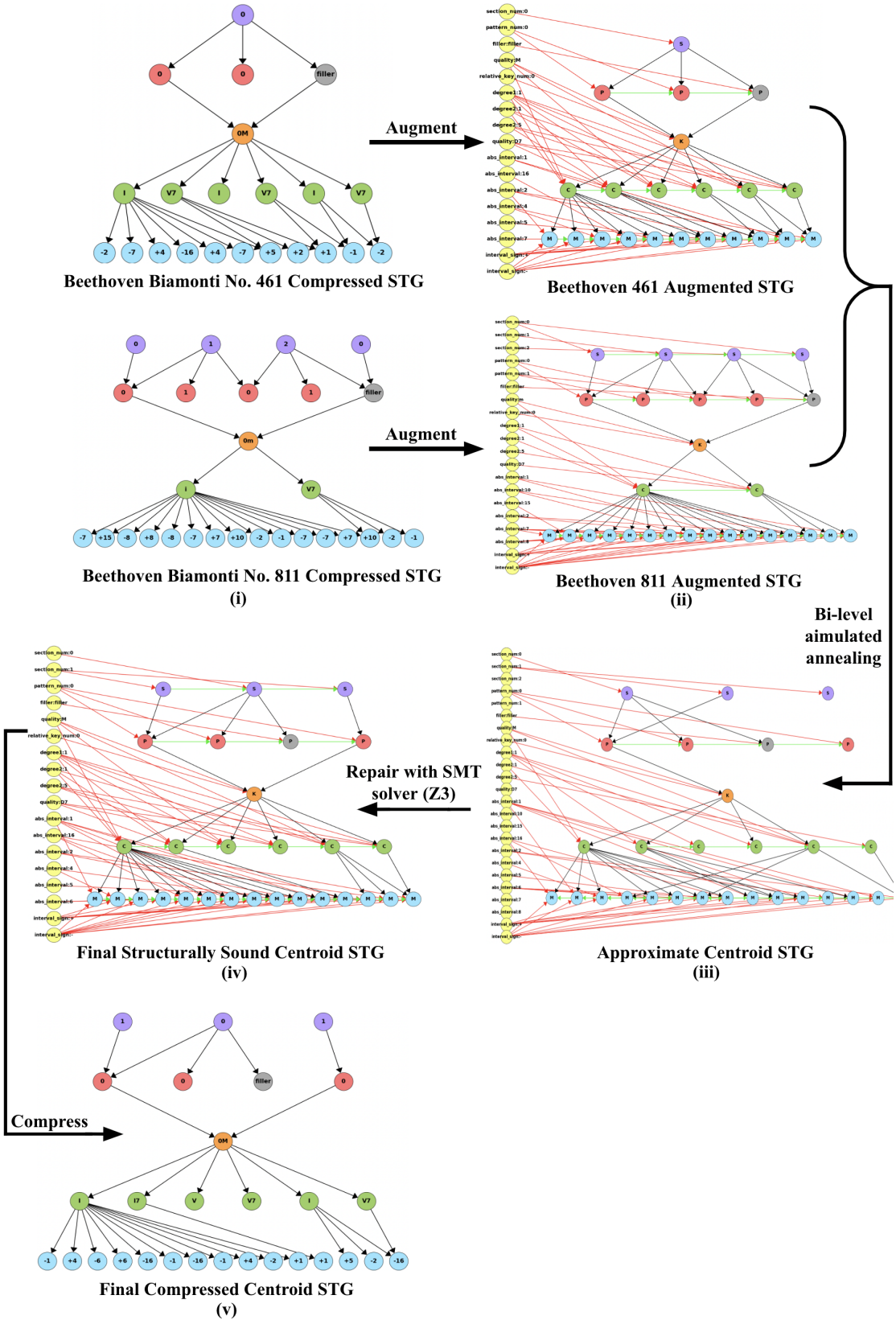

Figure 6: Example centroid derivation. In each STG, the segmentation nodes are purple, the motif nodes are red, the filler nodes are gray, the key nodes are orange, the chord nodes are green, and the melody nodes are blue. On the augmented STGs in (ii)-(iv), prototype nodes are to the left in yellow, prototype-instance edges are red, and intra-level linear chain edges are in green. Table 6 explicates the prototype node labels in more detail. Refer to Figure 4 for a zoomed-in view of the prototype nodes.