NEO-RIEMANNIAN CYCLE DETECTION WITH WEIGHTED FINITE-STATE TRANSDUCERS

Jonathan Bragg
Harvard University
jbragg@post.harvard.edu

Elaine Chew
Queen Mary, University of London
elaine.chew@eecs.qmul.ac.uk

Stuart Shieber
Harvard University
shieber@seas.harvard.edu

ABSTRACT

This paper proposes a finite-state model for detecting harmonic cycles as described by neo-Riemannian theorists. Given a string of triads representing a harmonic analysis of a piece, the task is to identify and label all substrings corresponding to these cycles with high accuracy. The solution method uses a noisy channel model implemented with weighted finite-state transducers. On a dataset of four works by Franz Schubert, our model predicted cycles in the same regions as cycles in the ground truth with a precision of 0.18 and a recall of 1.0. The recalled cycles had an average edit distance of 3.2 insertions or deletions from the ground truth cycles, which average 6.4 labeled triads in length. We suggest ways in which our model could be used to contribute to current work in music theory, and be generalized to other music pattern-finding applications.

1. INTRODUCTION

Though significant attention has been devoted to segmentation and labeling algorithms for discovering chords [14, 16, 19] and keys [4, 16, 18] in music scores, little work has been done on automating higher-level music analysis. One reason for the small body of research on this topic is that such analysis is highly subjective and relies heavily on musical intuition. Another reason is that there are numerous methods of analysis, which are often best suited to a particular corpus of music. We take a step toward bridging this gap between labeling and higher-level analysis by tackling the problem of finding neo-Riemannian cycles in chord sequences using a finite-state approach.

Neo-Riemannian music theory [17] posits that harmonies are related by means of transformations, rather than a common tonic. The theory defines three primary transformations P, L, and R that operate over the set of 24 major and minor triads (assuming enharmonic equivalence). Each transformation involves two triads that share two common tones. P transforms a triad to its parallel major or minor triad, L transforms a major triad to a minor triad whose root is four semitones higher (and vice-versa), and R transforms a triad to its relative major or minor triad. A cycle is generated by obtaining a triad, and repeatedly applying an identical permutation of either LP, RP, LRP, or LR at least until the originating triad is reached again. These cycles partition the harmonic space and give structure to certain musical works.

When neo-Riemannian theorists analyze a musical work, they locate a passage and identify harmonies that "participate" in a cycle. There are several motivations for automating this process. The first is to attempt to formalize the task, and in the process arrive at a more rigorous definition and understanding of what constitutes a cycle—and by extension what musical judgements are made during an analysis. The second is to facilitate a more comprehensive study of these cycles than currently exists [3]. Computer-aided analysis could provide a critique of the theory itself, as well as shed light on other music theoretic issues.

The existence of insertions and deletions presents challenges to accurately finding neo-Riemannian cycles. Suppose T_n is the composition of n transformations along a cycle. In theory, a cycle consists of a sequence of triads, such that each successive triad is generated by a single T_1 transformation. In practice, inserted harmonies intermix with the triads that participate in the theoretical cycle; and, triads in the theoretical cycle can be missing from the observable cycle due to the use of compound operations (T_n, where $n > 1$), or because the cycle is incomplete. On the surface, this problem may appear best solved by string matching algorithms. Approximate string matching algorithms [12] can handle insertions and deletions, and some methods have been developed to search for multiple strings [2]. The main problem with this approach is the representation of the search strings. LP cycles, for instance, consist of all strings beginning with $LPLPLP$ or $PLPLPL$ and continuing in like fashion, of which there are many. LP cycles alone partition the set of triads into four distinct cycles, each of which has six distinct originating triads and two directions of motion.

In contrast, a finite-state model facilitates the concise encoding of a cycle using transformations. It also enables us to represent transformational music theory in a visual and intuitive way. Specifically, we propose a noisy channel model to represent the task of finding an intended message (a cycle) given an observation sequence (of chords). Our implementation of the model uses weighted finite-state transducers (WFSTs). [15] describes this method, as applied to the realm of speech recognition.

Finite-state transducers (FSTs) are used extensively in language and speech processing [9], with potential applications to music. WFSTs, which are used to represent probabilistic finite-state machines in speech processing [11], could be used similarly in audio music processing. [10] uses WFSTs in the task of audio music identification as both an acoustic model and a compact language model. Drawing on efforts in language processing that implement the noisy channel model with WFSTs [13], our model is a novel application of this technique to symbolic music analysis.

The remainder of the paper is organized as follows. In Section 2, we formalize the problem statement and present the noisy channel model. In Section 3, we describe the input data, as well as the training and evaluation methods for our model. Finally, in Section 4 and Section 5, we present the results of our experiment and discuss our conclusions.

2. THE MODEL

Our goal is to design a system that will accurately identify and label all strings of harmonies corresponding to neo-Riemannian cycles in a music score. The input to the system is a string of triad labels representing a harmonic analysis, and the desired output is a version of that analysis with all musically salient cycles demarcated and labeled.

2.1 Problem Statement

Let Σ_1 be the alphabet consisting of symbols representing the 24 enharmonically distinct major and minor triads, and let $\Sigma_2 = \{P, L, R\}$, the alphabet of basic neo-Riemannian transformations. Also let $\Sigma_3 = \{[,]\}$, an alphabet of special demarcation symbols outside of Σ_1 and Σ_2. Now, suppose w is a string of symbols in Σ_1, corresponding to a harmonic analysis of a music score. The task is to identify exactly the substrings of w that correspond to neo-Riemannian cycles. These cycles should be labeled with the corresponding transformations from Σ_2 and bounded by symbols from Σ_3.

2.2 Noisy Channel Model

We implement the proposed noisy channel model with a cascade of WFSTs. Each component of the noisy channel model—a theory model, a noisy channel, and an observation sequence—is encoded as an FST. For simplicity of implementation, we reverse the direction of the model. Our reverse implementation is equivalent to the formal definition due to the closure of FSTs under inversion.

Our implementation is the composition

$$Score \circ ScoreEdit \circ Cycles$$

of FSTs representing chords in the observation sequence, chord edits in the noisy channel, and a model of (theoretical) cycles, respectively. We use the OpenFst library [1] implementation of FSTs and the Viterbi algorithm with the tropical semiring to calculate the path of lowest cost from $Score$ to $Cycles$. This scheme is appropriate to our transitions, which use weights rather than probabilities.

2.2.1 Score

$Score$ is the FST over Σ_1 that represents the observation sequence. As shown in Figure 1, $Score$ accepts and outputs exactly the string corresponding to our input data with no penalty. Its construction is simple to automate, since each transition from the start state to the final state corresponds to a triad in the input (in order). While we have not used this capability, our model can accommodate multiple weighted analyses of a piece, as shown in Figure 2.

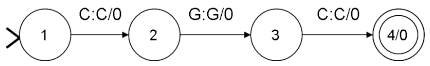

Figure 1. The $Score$ FST representing the score "C G C."

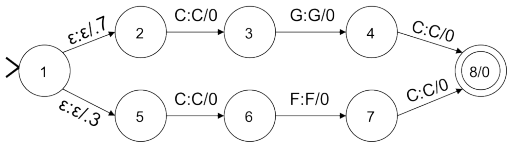

Figure 2. An FST representing a probabilistic encoding of two possible analyses of a hypothetical score.

2.2.2 ScoreEdit

$ScoreEdit$ is the FST that represents the noisy channel (in reverse). It transduces from Σ_1 to $\Sigma_1 \cup \Sigma_3$ and is defined as

$$ScoreEdit = AddBrackets \circ TriadsEdit, \qquad (1)$$

where $AddBrackets$ and $TriadsEdit$ are two smaller FSTs described below.

$AddBrackets$, shown in Figure 3, is a formatting step that demarcates cycles by inserting non-overlapping pairs of brackets into the score. In order to prevent an excessive number of cycles, we associate a cost B with the insertion

of a bracket pair, denoted $\epsilon : [\ / \ B$, meaning "Do not read an input chord. Add a bracket, at cost B." A transition labeled $\Sigma_1 : \Sigma_1 \ / \ 0$ is shorthand for all possible transitions labeled $\sigma_i : \sigma_i \ / \ 0$ such that $\sigma_i \in \Sigma_1$.

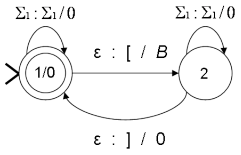

Figure 3. The *AddBrackets* FST.

Bracketing cycles in this way enables *TriadsEdit* to perform edits on the score that are sensitive to cycle boundaries. *TriadsEdit* operates over $\Sigma_1 \cup \Sigma_3$ and is defined as

$$\begin{aligned} TriadsEdit = \ & OutsideEdit \cdot (OpenBracket \\ & \cdot InsideEdit \cdot ClosedBracket \\ & \cdot OutsideEdit)^*, \end{aligned} \tag{2}$$

where *OpenBracket* and *ClosedBracket* are simple two-state FSTs that recognize the languages $\{[\}$ and $\{]\}$, respectively. As shown in Figure 4, *OutsideEdit* is a single-state FST over Σ_1 that deletes any number of triads (with cost X), and $InsideEdit$ is a single-state FST over Σ_1 that deletes, inserts, and reads any number of triads (with costs D, I, and 0, respectively). By construction of Equation (2), *OutsideEdit* operates only outside of cycles, *InsideEdit* operates only inside cycles, and zero or more cycles can occur anywhere in the score. We describe a method of training these weights (costs) in Section 3.2.

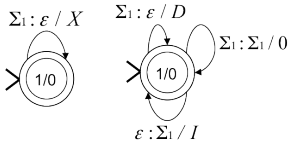

Figure 4. The *OutsideEdit* (left) and *InsideEdit* (right) FSTs.

2.2.3 Cycles

Cycles is the FST from $\Sigma_1 \cup \Sigma_3$ to $\Sigma_2 \cup \Sigma_3$. It transduces neo-Riemannian transformations from the cycles and is defined as

$$\begin{aligned} Cycles = \ & (OpenBracket \cdot Map \cdot ClosedBracket)^* \\ & \circ (OpenBracket \cdot Definitions \\ & \cdot ClosedBracket)^*, \end{aligned} \tag{3}$$

where *Map* and *Definitions* are the FSTs described below.

Map transduces from Σ_1 to Σ_2 and converts triads into transformations. It has a start state with transitions to each of the 24 other states corresponding to the major and minor triads. Each state corresponding to a triad is a final state, and has outgoing transitions to three other states according to P, L, and R transformations. Whenever *Map* in the start state reads a triad corresponding to a particular state, it moves to that state and outputs ϵ (with cost 0). From there, it is able to read successive triads and output the appropriate transformation symbols. For clarity, Figure 5 shows only a portion of *Map* corresponding to an LRP cycle, which contains 6 out of the 24 possible triads.

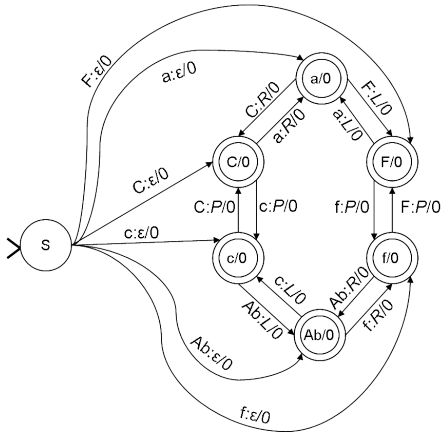

Figure 5. The *Map* FST (abbreviated).

Definitions is the FST over Σ_2 that recognizes any defined neo-Riemannian cycle. By construction, Equation (3) ensures that one of those cycles occurs within each set of brackets. *Definitions* is the union of all FSTs that represent a desired cycle, like the one shown in Figure 6.

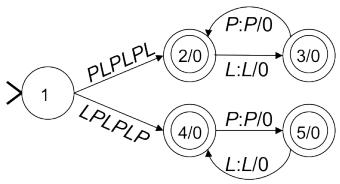

Figure 6. The *LP Cycle* FST. Each transition exiting the start state is shorthand for the transitions and intermediary (non-final) states necessary to transduce the labeled substring to itself with zero weight.

2.3 Generalizability

Our model is highly generalizable, and could be adapted to recognize various properties in a variety of music-theoretic systems. One could define new edit operations by modifying

ScoreEdit, incorporate other types of harmonies [6, 7] or transformations [5,8], or change *Map* to accommodate other conceptions of harmonic distance [20]. One could envision using our model to detect cycles in other music features such as rhythm (where the symbols might be durations rather than neo-Riemannian transformations), and patterns other than cycles.

3. EXPERIMENT

3.1 Input Data

We were able to obtain only a small quantity of input data from scores in the desired corpus of late Romantic music scores, to which neo-Riemannian analysis is typically applied. Neither a dataset of harmonic analyses, nor a reliable way of automatically converting music scores into analyses is presently available. Thus, the first author performed all analyses manually prior to the automated analysis. Seventh and other extended chords were reduced to their underlying triads, and vertical sonorities without a prominent major or minor triad identity were ignored.

Our input data are selections from four works by Franz Schubert in which [17] identifies LP and RP cycles. [17] analyzes two LP cycles in the exposition of the first movement of the A major Piano Sonata, D. 959, one LP cycle in the fourth movement of the G major Piano Sonata, D. 894, one LP cycle in the coda of the first movement of the E-flat major Piano Trio, D. 929, and one RP cycle in the first movement of the C major String Quintet, D. 956. Since the focus of this experiment is LP and RP cycles, we define the *Definitions* FST to recognize either one. Given the small size of our dataset, it was not necessary to perform the usual determinization and minimization algorithms to make the FSTs in our model time- and space-efficient, respectively.

In order to describe and classify the cycles that comprise our ground truth, we identify properties of cycles that are visible to our model. Let p be the number of triads in an observable cycle that are labeled with transformations, let o be the number of triads that are not labeled (insertions), and let $n = o+p$ be the overall length. Also, let m be the number of deletions, and let l be the length of the shortest complete theoretical cycle of the type being labeled (e.g. $l = 7$ for LP cycles). Note that $p + m = l$, except for extended cycles, where $p + m > l$. Table 1 shows o, m, p, and l for each of the cycles in our input data.

We also calculate two quantities in Table 1 that help us to classify cycles. $\frac{o}{o+p}$ is the proportion of insertions relative to the observable length, and $\frac{m}{m+p}$ is the proportion of deletions relative to the length of the corresponding theoretical cycle. We will use these two quantities, also graphed in Figure 8, to explain our results.

Piece	Measures	o	m	p	l	$\frac{o}{o+p}$	$\frac{m}{m+p}$
D. 959 (ex. 1)	28–36	9	4	5	7	0.64	0.44
D. 959 (ex. 2)	82–103	24	0	9	7	0.73	0
D. 894	154–160	21	3	4	7	0.84	0.43
D. 956	233–250	9	2	7	9	0.56	0.22
D. 929	585–612	9	0	7	7	0.56	0

Table 1. Cycles in the ground truth and their properties.

3.2 Training Method

Training our model consists of setting four parameters: B, D, X, and I, which are the costs of bracketing cycles, deleting chords inside cycles, deleting chords outside of cycles, and inserting chords, respectively (described in Section 2.2). While systems can be trained with musically-informed rules [19], we calculate weights empirically. Our method involves setting up a system of linear inequalities by determining the behavior of our system over isolated strings of n triads.

To privilege labeling a cycle of n triads over deletion, we use equations of the form

$$B + oD + mI < nX. \tag{4}$$

To privilege deletion, we would simply reverse the inequality. We generate instances of Equation (4) from a ground truth labeling of a score by selecting each cycle and calculating o, m, and n. In order to prevent our system from arbitrarily extending cycles it labels, we also require that

$$D > X. \tag{5}$$

We solve the resulting system by minimizing the objective function $B + D + I + X$.

3.3 Evaluation

The desired performance metric should measure the success of both segmentation and labeling of cycles.

We propose an evaluation method that uses global string alignment applied separately to each region in the score with one or more overlapping cycles in either the ground truth or the prediction. Since a string of transformations does not uniquely determine the underlying triads, we do not compare those strings. Instead, we calculate the edit distance between the string of triads labeled with transformations (i.e. not insertions) in the prediction with the corresponding string in the ground truth. Allowable edit distance operations are insertion and deletion, like in our model. If a cycle does not exist in one labeling, the edit distance is simply the cost of deleting all symbols in the other string. This metric has the property that segmentation errors are proportional to p and not o; it is a measure of divergence in transformational content rather than overall observable content.

Piece	1	2	3	4	5	6	7	8	9	10	11	S_n	S_p	S_t
D. 959	5	**4**	6	**0**	5							4	16	20
D. 894	8	**10**	6	8								10	22	32
D. 956	6	5	9	7	**0**	7	6	5				0	45	45
D. 929	6	5	6	7	8	7	7	8	4	7	**2**	2	65	67

Table 2. Alignment costs for each piece, broken down by region. Bold formatting indicates that the region contains a cycle in the ground truth.

The evaluation score S_t of a prediction is equal to the sum of all edit distances calculated as just described, i.e. $S_t = S_n + S_p$, where S_n is the sum of all edit distance operations on regions with a cycle in the ground truth, and S_p is likewise defined on all other aligned regions. S_n and S_p measure in some sense the amount of "false-negativeness" and "false-positiveness," respectively, in a prediction.

We use leave-one-out cross-validation on our four pieces of input data. Training for validation on D. 959, D. 956, and D. 929 each yielded weights $I = 1$, $B = 1$, $D = 1.0065$, and $X = 1.0055$, and training for validation on D. 894 yielded weights $I = 1$, $B = 1$, $D = 1.003$, and $X = 1.002$. Table 2 shows a breakdown of performance by aligned region for each score.

4. RESULTS

In our experiment, we used the cycles analyzed by [17] as our "ground truth." If we define successful retrieval of a cycle in the ground truth as prediction of a cycle in the same aligned region, our model achieved precision and recall scores of 0.18 and 1.0. (The model predicted a cycle in every aligned region containing a cycle in the ground truth.) The cycles recalled from the ground truth, on average, had length $p = 6.4$ and alignment score 3.2.

Our choice of ground truth cycles impacted our precision score and led to many predicted cycles in regions not analyzed. Viewed as strings of harmonies, these predicted cycles are difficult to distinguish from cycles in the ground truth. In particular, our model predicted an RP cycle in measures 304–329 (aligned region 7) of D. 929 with dimensions $o = 8$, $m = 2$, $p = 7$, and $l = 9$, which almost exactly match the dimensions of the ground truth RP cycle in D. 956 (see Table 1). We arrive at the conclusion that either the ground truth is incomplete, or that other factors affect theorists' decisions on what constitutes a cycle.

Our model also labels cycles on a more detailed level than is often done in music analysis. In practice, theorists often describe transformations acting on a cluster of chords with a prominent harmonic identity, rather than a particular chord with that identity. By contrast, our model always labels specific chords with transformations. Our evaluation measure does not penalize this type of over-specification. Aligned region 5 of D. 956, which received one of two perfect alignment scores, illustrates this point. In translating the analysis in [17] to the ground truth labeling, the first author selected the second D major chord shown in Figure 7 for participation in the theoretical cycle based on cadential and inversional information in the score. Our model selected the first D major chord instead, but was not penalized by construction of our evaluation method.

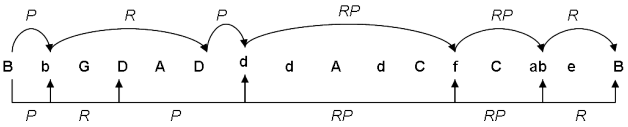

Figure 7. Aligned region 5 of D. 956 (mm. 233–250), with ground truth labels (curved connectors) and predicted labels (elbow connectors).

While our model predicted a cycle in each aligned region containing a cycle in the ground truth, misalignments of varying severity also occurred. The predicted cycles in aligned region 11 of D. 929, aligned region 2 of D. 959, and aligned region 2 of D. 894 received increasingly large evaluation scores. These increasing scores reflect the costs of identifying an extended cycle, a cycle with the desired harmonic content but opposite direction, and a cycle with altogether different harmonic content, respectively.

In order to understand why these cycles posed challenges to our model, consider Figure 8. Distance from the origin correlates with the alignment scores of these three cycles. In addition, there seems to be a direct link between distance from the x-axis (corresponding to the relative number of deletions) and poor performance. Tellingly, the three cycles with the best scores (aligned region 4 of D. 959, aligned region 5 of D. 956, and aligned region 11 of D. 929) are located on or near the x-axis, but not particularly near the y-axis, suggesting that the model is able to handle many inserted triads, so long as there are few deletions. The two remaining cycles in the figure, located furthest from the x-axis, were more costly to align. Each consists of strictly T_2 transformations, resulting in many deletions. The finite-state model is not in general well-equipped to reward regularity in patterns, and in this case was not able to recognize regularity of motion within a cycle.

To view the complete set of musical excerpts and extracted harmonic analyses, please visit `http://www.jonathanbragg.com/ismir2011`.

5. CONCLUSION

This paper presents the essential design and performance of a finite-state approach to harmonic cycle detection. The model performed well on the task at hand: with access to

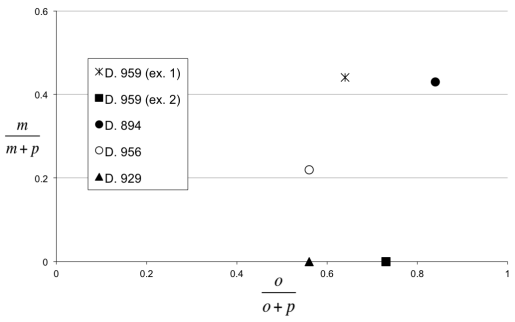

Figure 8. Plot of proportion of deletions vs. proportion of insertions (data from Table 1).

very little music feature data, it predicted all cycles in the ground truth, some with very high accuracy, and suggested other potentially viable cycles. As more harmonic analysis data becomes available, it will be possible to do more extensive testing of the model, and to incorporate other features. In its current form, the model could be used as a tool for theorists, to propose potential cycles which might be analyzed and catalogued, and ultimately contribute to a better understanding of cycles and neo-Riemannian theory. This approach is highly generalizable and can be applied to other kinds of pattern matching in music.

6. ACKNOWLEDGEMENTS

This work was supported in part by the Harvard College Program for Research in Science and Engineering and NSF Grant No. 0347988. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect those of Harvard University or NSF.

7. REFERENCES

[1] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFst: A general and efficient weighted finite-state transducer library. In *CIAA 2007*, volume 4783 of *LNCS*, pages 11–23. Springer, 2007. `http://www.openfst.org`.

[2] R. Baeza-Yates and G. Navarro. Multiple approximate string matching. In *WADS 1997*, volume 1272 of *LNCS*, pages 174–184. Springer, 1997.

[3] M. Bribitzer-Stull. The Ab-C-E complex: The origin and function of chromatic major third collections in nineteenth-century music. *Music Theory Spectrum*, 28(2):167–190, 2006.

[4] E. Chew. Regards on two regards by Messiaen: Post-tonal music segmentation using pitch context distances in the spiral array. *Journal of New Music Research*, 34(4):341–354, 2005.

[5] R. Cohn. Square dances with cubes. *Journal of Music Theory*, 42(2):283–296, 1998.

[6] E. Gollin. Some aspects of three-dimensional "tonnetze". *Journal of Music Theory*, 42(2):195–206, 1998.

[7] J. Hook. Uniform triadic transformations. *Journal of Music Theory*, 46(1):57–126, 2002.

[8] B. Hyer. Reimag (in) ing Riemann. *Journal of Music Theory*, 39(1):101–138, 1995.

[9] M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[10] M. Mohri, P. Moreno, and E. Weinstein. Efficient and robust music identification with weighted finite-state transducers. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(1):197–207, 2010.

[11] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[12] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[13] R. Nelken and S. Shieber. Arabic diacritization using weighted finite-state transducers. *Computational Approaches to Semitic Languages*, 8:79, 2005.

[14] B. Pardo and W. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, 2002.

[15] F. Pereira and M. Riley. Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing*, pages 431–453. MIT Press, 1996.

[16] C. Raphael and J. Stoddard. Functional harmonic analysis using probabilistic models. *Computer Music Journal*, 28(3):45–52, 2004.

[17] M. Siciliano. *Neo-Riemannian Transformations and the Harmony of Franz Schubert*. PhD thesis, University of Chicago, 2002.

[18] D. Temperley. *Music and Probability*. MIT Press, Cambridge, Massachusetts, 2007.

[19] D. Temperley and D. Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, 1999.

[20] D. Tymoczko. Three conceptions of musical distance. In *Mathematics and Computation in Music*, volume 38 of *CCIS*, pages 258–272. Springer, 2009.