Algorithms for Chordal Analysis

Bryan Pardo and William P. Birmingham
Artificial Intelligence Laboratory
Electrical Engineering and Computer Science Department
The University of Michigan
110 ATL Building, 1101 Beal Ave.
Ann Arbor, MI 48109-2110, USA
bryanp@umich.edu
wpb@eecs.umich.edu

The harmonic analysis of tonal music by computer is an important area of interest in the computer music research community. While the problem is interesting in its own right, the ability to parse and use chords and harmonies in a tonal composition also adds an important dimension to a computer agent's ability to manipulate musical material. Automated arranging, accompaniment, and phrasing are all aided by an understanding of the chordal structure of a piece.

Although musicians analyze tonal music of all types, it is difficult to create algorithms that have this generality. In this article, we describe a set of algorithms for harmonic analysis that make minimal use of stylistic and contextual cues, such as metric strength, harmonic context, and known stylistic constraints. The system described in this article is purposefully simple in its approach. This allows us to specify its workings to the point where the work may be duplicated and extended by others. The system is designed both to explore basic computational properties of algorithms to solve the task and to provide a baseline against which other systems can be measured. In this way, it is possible to measure the change in performance introduced by use of such things as tonal context, voice-leading rules, and metrical information.

We divide harmonic analysis into two tasks. *Segmentation* is the task of splitting the music into appropriate chunks (segments) for analysis. As a default, we assume segmentation takes place in the time dimension. *Labeling* is the task of giving each segment the proper quality and root pitch. Figure 1 shows a measure of music partitioned into labeled segments.

The algorithms we describe in this article perform both segmentation and labeling. Previous research in the area of automated harmonic analysis of music (Winograd 1968; Smoliar 1980; Laden and Keefe 1989; Ebcioglu 1992; Maxwell 1992; Widmer 1992; Smaill et al. 1993; Taube 1999), with the notable exception of recent work by Temperley and Sleator (1999), has either avoided the issue of segmentation by taking already segmented input, or has been unclear about how segmentation is done. Temperley and Sleator describe a clear and efficient approach.

While Temperley and Sleator do address the segmentation problem, they only return the root name (rather than the chord quality) and do not give any statistical analysis of the performance of their system when compared to an outside measure. To our knowledge, no researchers have published statistical performance results on a system designed to analyze tonal music. The lack of such measures in the literature makes comparisons among systems difficult.

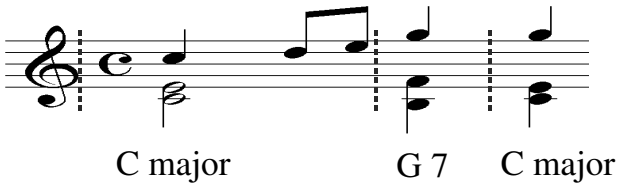
We present an empirical evaluation of the algorithms in this article, using a corpus of 45 excerpts of tonal music compiled by David Temperley, from the Kostka and Payne music theory textbook (Kostka and Payne 1984), hereafter referred to as the KP corpus. (The appendix lists each piece used in this corpus.) The analyses provided by our system are compared to an answer key based on the analyses in the teacher's edition of the textbook using a grading metric described later in this article. Both statistical results and analyses of individual excerpts are provided, and classes of error are tabulated and reported.

Segment Labeling

We define two major tasks in harmonic analysis: segmentation (i.e., identifying those places in the music where the harmony changes) and labeling (i.e., giving each segment the proper quality and

Computer Music Journal, 26:2, pp. 27–49, Summer 2002

Figure 1. Music partitioned into labeled segments.



root). If the right segmentation has already been determined, labels must still be determined. This section describes an approach to labeling segments that compares the collection of notes in a segment to templates describing common chords.

This approach to labeling a segment uses a fixed number of steps irrespective of the number of notes in the set and has the benefit of generating a numerical measure of proximity to the nearest template in the system's vocabulary. These two properties have important ramifications that allow use of the segment-labeling algorithm as a key component in the segmentation process described in a later section. However, for the purposes of describing segment labeling here, we assume segmented input.

The chromatic scale and its twelve pitch classes are basic elements associated with most tonal music. The common labels for the pitch classes, along with their integer equivalents, are given in Table 1. Using the integer representations of the pitch classes and modulo-12 arithmetic, structures such as chords and scales can be represented as n-tuples. These tuples correspond to positive displacements in the space of pitch classes in relation to a root pitch class. The tuples imply templates that we use to find musical structures. These templates are closely related to sets used in atonal set theory (Forte 1973), the chromagram (Wakefield 1999), and the work of Ulrich (1977).

An example of the template representations is the following. Given a root (pitch) class r, the tuple $\langle 0, 4, 7 \rangle$ represents the pitch-class relations to r embodied in a major triad. Letting $r = 7$ results in a chord given by $\text{mod}12(r+0, r+4, r+7) = \{7,11,2\}$. From Table 1, it is easy to verify that this corresponds to {G, B, D}, the pitch classes in the G major triad.

The system described in this article used a set of 72 templates derived from six template classes. Each class has twelve member templates, one for each root pitch class. These are generated in a manner identical to that of the example G major triad described in the previous paragraph. Table 2 lists the template classes used in the experiments reported in this article. Chordal template categories were selected based on their frequency of occurrence in the harmonic analyses in the answer key to the KP corpus of 45 musical excerpts. All chord qualities accounting for at least 2% (when rounded) of the chord labels in the corpus were used. This was done to limit the negative impact of extremely rare chord templates on the performance of the system.

It is simple to add new templates. Note that nothing in this approach limits templates to triadic structures. Pentatonic scales and quartal constructions, for instance, can be identified simply by introducing a template for the structure in question and establishing a rule for resolving ties between the templates when more than one is applicable. It is also important to note that templates are not heuristic: the pitch relationships described in the template must hold for the tonal structure it models, by definition of the structures.

Before describing the scoring mechanism for determining which template best matches a collection of notes, we must introduce some terminology. Harmonic change can only occur when at least one note begins or ends. A *partition point* occurs where the set of pitches currently sounding in the music changes by the onset or off-

Table 1. Pitch class number and name correspondences

C	*C-sharp D-flat*	*D*	*D-sharp E-flat*	*E*	*F*	*F-sharp G-flat*	*G*	*G-sharp A-flat*	*A*	*A-sharp B-flat*	*B*
0	1	2	3	4	5	6	7	8	9	10	11

Figure 2. Segments and Partition Points.

Figure 3. (Bottom of page) Calculating the score of a template.

Table 2. The template classes used by our system

Chord Name	*Proportion of KP corpus*	*Template Representation*
Major Triad	43.6%	⟨0 4 7⟩
Dom7 (Major-Minor)	21.9%	⟨0 4 7 10⟩
Minor Triad	19.4%	⟨0 3 7⟩
Fully Diminished 7th	4.4%	⟨0 3 6 9⟩
Half Diminished 7th	3.7%	⟨0 3 6 10⟩
Diminished Triad	1.8%	⟨0 3 6⟩

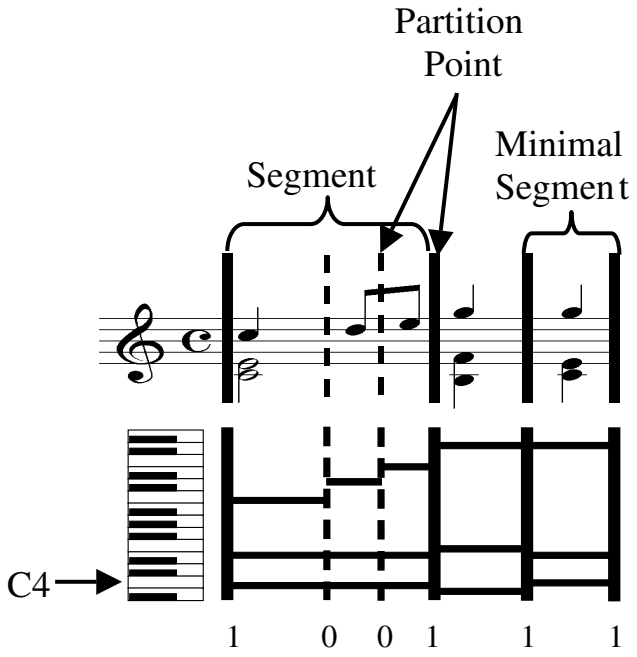
set of one or more notes. A *segment* is a contiguous interval between two partition points. A *minimal segment* is the interval between two sequential partition points. Figure 2 illustrates these concepts.

Given a segment containing a collection of notes, the score for a particular template is calculated from the steps in Figure 3.

As an example, let us calculate the score for an A minor triad template on the segment comprising the first two beats of the measure in Figure 2. The template for the A minor triad is found by adding 9 (the pitch class of A) to the values in the template class for a minor triad, ⟨0, 3, 7⟩, and taking modulo 12 of the resulting values. This results in the template ⟨9, 0, 4⟩, or {A, C, E}. We may now follow the steps in Figure 3.

There are five notes in the first segment, {C4, E4, C5, D5, E5}, whose weights are 3, 3, 1, 1, and 1, respectively. Each C and E in the segment matches a template element, so we add their weights:

$$P = Weight(C4) + Weight(E4) + Weight(C5) + Weight(E5) = 3 + 3 + 1 + 1 = 8$$



The pitch D5 is the only one that does not match any template, so we have

$$N = Weight(D5) = 1$$

One template element (for the pitch A) was not found in the segment. To summarize, we have

$$M = 1$$
$$S = P - (M + N) = 8 - (1 + 1) = 6$$

Thus, the final score for the A minor triad template of this segment is 6.

1. Determine the weight of each note in the segment by counting the number of minimal segments in which the note is present between the start and end of the current segment.
2. Sum the weights of the notes whose pitch class matches a template element. Call this the Positive Evidence, P.
3. Sum the weights of the notes not matching any template element, Call this Negative Evidence, N.
4. Sum the count of template elements not matched by any note. Call these the Misses, M.
5. Calculate the score for a template, S, with the formula $S = P - (M + N)$

Figure 4. Tie breaking between templates.

1. ROOT WEIGHT: Choose the template whose root pitch class has the greatest weight of notes present in the segment.
2. PRIOR PROBABILITY: Choose the template with higher prior probability of occurrence (see Table 2).
3. DIM7 RESOLUTION: If all top templates are fully diminished 7^{th} chords, select the template whose root is one half-step below the root of the top scoring template in the following segment (contextual).

In this scoring mechanism, the number of minimal segments a note spans determines its weight. Note that this does not directly correlate to the absolute duration of the note or to its notated rhythmic value. Rather, it correlates to the number of changes in the music the note overlaps. Thus, a quarter note in the first beat of Figure 2 spans a single minimal segment, as do each of the eighth notes in the following beat.

One could imagine any number of note-weighting methods involving actual duration, volume (encoded as MIDI velocity), weighting the highest or lowest note differently, etc. We tried a number of ad hoc approaches to note weight involving these parameters but found that the approach based on minimal segments worked surprisingly well. Furthermore, this corresponds to our design philosophy, which emphasizes the use of the simplest approach wherever possible.

To label a segment, all 72 templates are scored, and the highest scoring template determines the segment label. When this is done for the first two beats of the measure in Figure 2, the C major triad ties with the A minor triad, and each receives a score of 6. If two templates generate the same score, they form an equivalence class, and the winner may be selected using any tie-resolution rule. Some tie-breaking rules we used with success in our system are shown in Figure 4. The rules are applied in the order they appear in Table 2. For our example, the first tie-breaking rule causes the segment to be labeled C major, since the pitch weight for C in the segment is 4, while that for A is 0.

The first two tie-breaking rules in Figure 4 are examples of context-free tie resolution. The third rule performs contextual tie breaking. These rules are not intended to be an exhaustive list; any number of rules may be envisioned involving tonal context, position of lowest note, etc. A later section of this article explores the effect these rules have on system performance.

Table 3 shows the label for every segment in Figure 2, using the templates, scoring, and tie-breaking rules described in this section. The vertical key determines the index number of the starting partition point for the segment. The horizontal key determines the ending partition point for the segment. The number beneath each chord label is the score that template received on the set of notes in the given segment. For example, the segment ⟨4, 6⟩ consists of the last two beats of the measure. The label best matching these two beats considered as an aggregate is G7, and it receives two points.

The approach described in this section labels and scores each segment in constant time given that the note weights for each segment have already been determined.

The minimal segments form the main diagonal of Table 3. Calculating the note weight for the minimal segment from MIDI input may be done in linear time with respect to the number of notes in a piece by maintaining a 12-element array, indexed by pitch-class number (0 through 11), representing the state of the music. Each element *i* gives the count of notes of pitch class *i* currently sounding. The array is updated as MIDI data are received, adding one to the appropriate element for a "note on" and subtracting one for a "note off." When a non-zero MIDI delta-time is encountered, this indicates the beginning of a new minimal segment, and the state is saved as a minimal segment before updating it with the new note event.

The time required to generate the set of minimal segments is linear with respect to the number of MIDI events in the input. This can be seen by noting that a fixed number of steps is taken for any "note on" or "note off" event. Each of these steps

Table 3. Segment scores for Figure 2

	2	*3*	*4*	*5*	*6*
1	C Major 2	C Major 3	C Major 6	C Major 6	C Major 9
2		C Major 0	C Major 3	C Major 3	C Major 6
3			C Major 2	C Major 2	C Major 5
4				G7 2	G7 2
5					C Major 3

requires a fixed amount of time. Thus, the bound on the complexity is *O(n)* (i.e., linear with respect to *n*), where *n* is the number of notes in the music.

Once the note weights for minimal segments have been computed, all other segment weights may be derived from them, taking a fixed number of steps per segment. This may be done recursively by noting that the weight of segment ⟨i, j⟩ is given by the sum of the weight of segment ⟨i, j–1⟩ and weight of segment ⟨i + 1, j⟩. (Pardo and Birmingham (2001) offer a detailed description of how to construct the full table of segments.) Now we turn to an analysis of the efficacy of the labeling approach described in this section.

Empirical Testing of the Segment-Labeling Algorithm

The segment-labeling algorithm is simple, considering only structures based on pitch class and relative durations of notes. No notion of tonal context, beat, absolute pitch height, or dynamics enters into the calculation. How well, then, does such an algorithm do in labeling segments on typical examples of tonal music?

Before presenting our experiment results, we describe the method we used to grade the labeling algorithm. Figure 5 shows grade calculation for our labeling system on a two-bar passage. The correct answer and the program's analysis are shown for each segment. The system receives one point for each minimal segment where its label exactly matches the answer key. If the program returns multiple answers and the correct answer is among them, the point is divided by the number of guesses taken. Segments containing answers not available to the system (such as ''German 6th'' or ''rest'') are not graded. These are indicated by a hyphen on the ''Points'' line.

This is a non-forgiving grading measure, as can be seen from Figure 5. The only way to achieve a perfect score is to label every minimal segment correctly and unambiguously. In the example, the system did not apply label tie-breaking and was unable to distinguish between C major and C minor in the third beat of the first measure, costing it roughly 6% of the final grade. The grading measure is also unforgiving of issues such as labeling a major triad with a passing seventh as a dominant seventh chord, as happens in the final beat of the first measure. We considered a scoring measure that would be more forgiving of such errors but decided that the clarity of this grading measure outweighs its drawbacks.

Using this grading method, we compared the chord labels assigned by our labeling method with those given in the answer key to the KP corpus. The corpus gives chord labels using standard Roman numeral notation and were translated by the first author to a format compatible with our labeling system. For example, C: I – vii^{o6} – I^{6} would become C major – B dim – C major.

For each excerpt, the segmentation was determined by the positions of the chord labels in the

Figure 5.Grading a labeling.

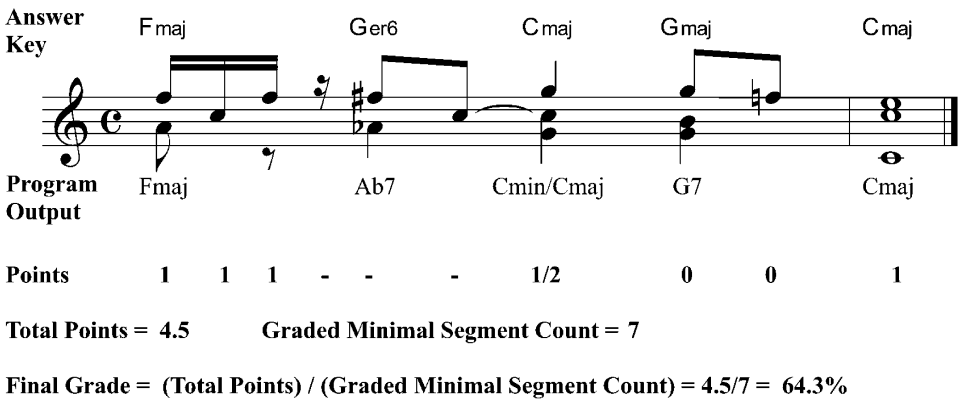


Table 4. Mean label scores by tie-breaking regime

Tie-Breaking Strategy	*Mean % of Piece with Multiple Labels*	*Mean Score %*	*Num Cases*	*Std Dev of Mean Score*
No tie breaking	9.45	84.54	45	13.00
Rule 1: Root Weight	4.27	85.65	45	11.77
Rule 2: Prior Probability	7.94	86.12	45	11.44
Rule 3: Dim 7 Resolution	7.53	86.11	45	13.03
Rules 1 and 3	2.22	86.97	45	10.83
Rules 1 and 2	3.74	87.22	45	11.78
Rules 2 and 3	6.03	87.80	45	11.28
Rules 1 and 2 and 3	1.68	88.66	45	10.72

answer key, with one segment per label, beginning where the label first appears and continuing until the next label appears. The note weights for each segment in an excerpt were passed to a function encoding the labeling method outlined in this article, and the label (or labels) returned was recorded. Once labels for all segments were determined, the resulting labeling was scored against the answer key using the method previously described.

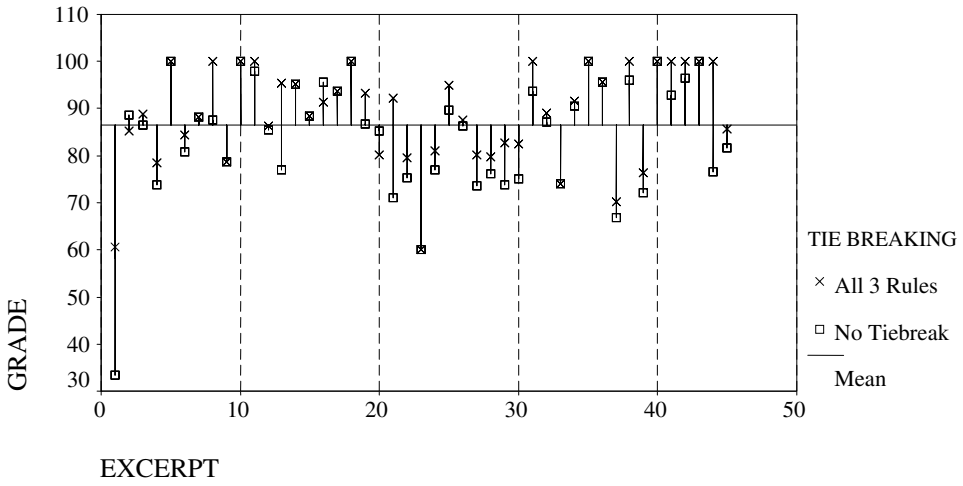
To explore the effects of the three tie-breaking rules in Figure 4, each piece in the corpus was labeled using eight methods: no tie breaking, each rule in isolation, every pair of rules, and all three of these. This resulted in a total of eight trials per piece. Table 4 shows mean grades for the entire corpus, broken down by tie-breaking regime.

As can be seen from the table, the algorithm without tie-breaking rules returns multiple labels, for an average of over 9% of the minimal segments in a piece. When all three rules are applied, this figure drops below 2%. This improves the mean grade from below 85% correct without tie breaking to nearly 89% correct with the use of all three tie-breaking rules.

Figure 6 compares the labeling algorithm's performance using all three tie-breaking rules against its performance without the use of tie breaking. The horizontal line is the mean of all cases shown in the figure. All trials for a single excerpt are aligned vertically along a spike line from the mean line.

As the figure shows, the number of perfect scores increases from six (with no tie breaking) to 13 (using all three rules). While this is a great improvement, the labeling method continues to return sub-optimal scores on 32 of the 45 excerpts. In addition, tie breaking actually lowered the labeling score on excerpts 2, 16, and 20. In these cases, a

Figure 6. Labeling Grades by Excerpt.



tie-breaking rule eliminated a correct template from consideration, lowering the final grade.

Figure 5 shows how this can occur. The third beat of measure 1 is missing the third of the chord, and two templates receive equal scores. If the tie-breaking rules from Figure 4 are applied, the system will select ''C major'' based on the relative frequency of major versus minor chords in the corpus as shown in Table 2. In the case of this example, this gives the correct result; however, had the passage been in a minor key, ''C major'' would have still been selected, and the result would have been incorrect.

It is instructive to look at individual labeling errors to determine their causes. The system made the most errors on Excerpt 1, the anonymous Minuet in G from the *Notebook for Anna Magdalena Bach*. Figure 7 shows this excerpt, with the answer key at the top of each system and the answers generated by the algorithm after tie breaking shown at the bottom of each system. Labels remain in effect from the point where they appear until the next label is encountered. Errors are boxed and labeled.

Figure 7 illustrates several types of errors. The worst error can be found in the third and eleventh measures. These measures are identical, and both are mislabeled with a very unlikely chord name, given the tonal context. The penultimate measure shows a place where there are exactly as many notes in a G major triad as in a D major triad. The system is unable to resolve the tie and returns both answers. Adding extra weight to the bass note or ruling out obvious passing tones from consideration could have resolved the errors in all three of these cases.

The remainder of the errors are on the excerpt are better in the sense that a human might choose to label things as the system has. The KP corpus answer key labels notes appearing to be a second inversion V (or V^7) chord between two I chords as a vii^{o6} chord. Thus, measures five and thirteen are labeled F-sharp diminished. Our system returns the label of the V^7 chord (i.e., D7) for these measures. Both choices serve the same serve the same functional role in this context. In the eighth measure, the system includes the C into the chord, resulting in a D7 label, while the answer key calls C a passing tone, resulting in a label of D major. Again, the same functional role is served by either chord.

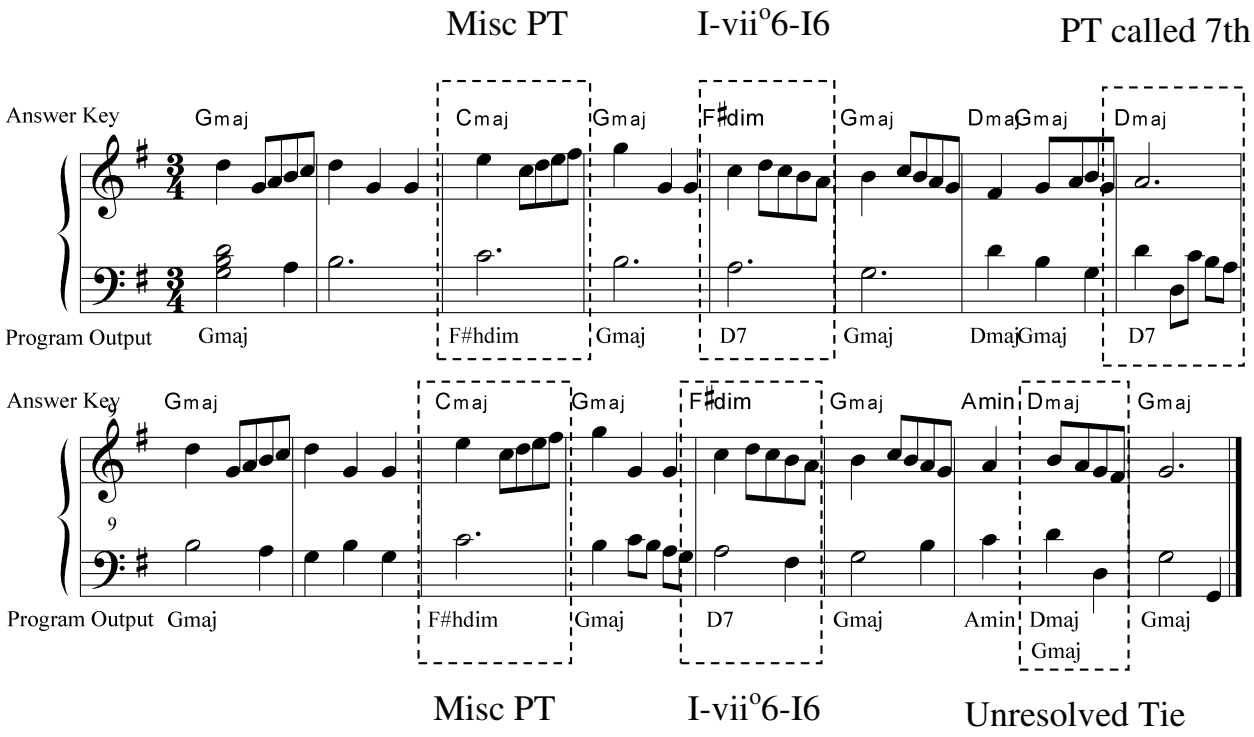
To gain a better understanding of what causes the remaining error when full tie breaking is enabled, each labeling error on every excerpt in the corpus was examined and categorized, resulting in the following classes.

Miscellaneous passing tone: An error was caused by the system's inability to discount passing tones from consideration.

Bad tiebreak: A tie-breaking rule eliminated the correct answer.

Disagree with answer key: We believe the answer key has an incorrect chord label.

Figure 7. Excerpt 1, anonymous Minuet in G, from the Notebook for Anna Magdalena Bach.



Harmonic context required: An incomplete chord, which can only be uniquely identified through tonal function in context, is misidentified.

I – vii^{o6} – I^{6}: The KP corpus answer key labels notes that appear to be a second inversion V (or V7) chord between two I chords as a vii^{o6} chord.

Performance variation: A timing variation in performance caused notes in the MIDI file to overlap in different combinations than indicated in the score, affecting the program's interpretation.

Miscellaneous unresolved tie: Miscellaneous tie between chord labels.

Pedal tone: The system incorrectly classifies a chord by including a pedal tone.

Passing tone called seventh by algorithm: The program labels a triad with a passing seventh in the answer key as a seventh chord.

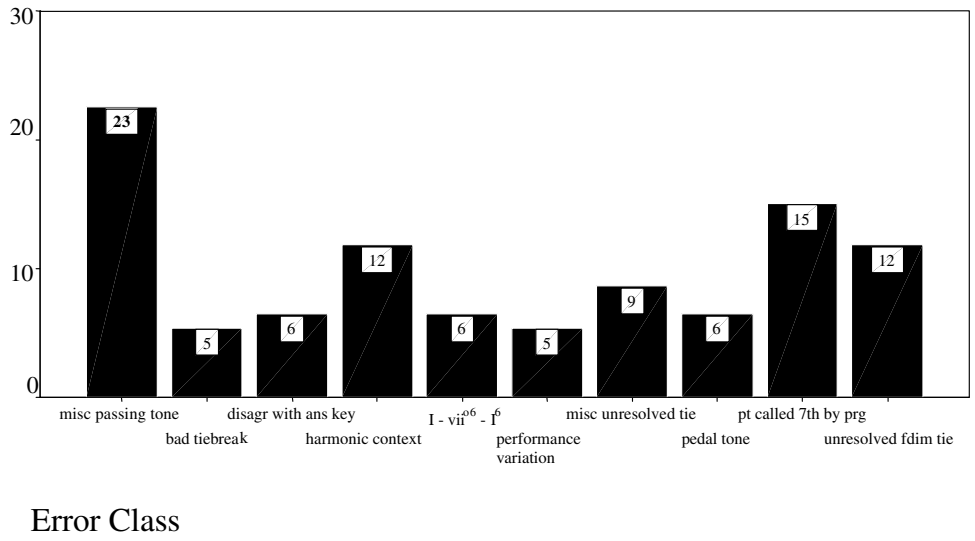
Unresolved fully diminished tie: Fully diminished seventh chord does not resolve by step (e.g., common-tone diminished chord), so our tie-resolution rule does not resolve a tie between the four possible roots for the chord.

These categories do not represent a serious attempt to develop an error taxonomy; rather, they are natural categories resulting from hand analysis of the data. Thus, some classes represent relatively general categories and others (such as "I-vii^{o6}-I^{6}") represent specific, frequently occurring errors. Error frequencies as a percentage of the total are shown in Figure 8.

Figure 8 suggests certain things about the labeling system. Perfect tie breaking can be expected to eliminate three of the error classes ("bad tie break," "miscellaneous unresolved tie," and "unresolved fully diminished tie"), for a total 26% of the errors. This is the maximum improvement that can be expected from improved tie breaking.

Specific idiomatic errors, such as "I – vii^{o6} – I^{6}," may be resolved by using a two-pass system. The first past would generate the initial chord names. The second pass could look for sequences of chord names that have the functional form "I – V$^{4}_{3}$ – I^{6}"

Figure 8. Error Class Frequencies on the KP Corpus.



and replace the middle chord with a diminished chord. System performance might be further improved by adding beat information to the note weights. This could reduce the ''miscellaneous passing tone'' and ''pedal-tone'' errors, although heavily syncopated music would still present a problem.

The current labeling system only considers harmonic context through the diminished chord tie-breaking resolution rule. This does not affect the raw label scores. Knowledge of adjacent chord labels might be used to adjust chord-label weights. This introduces a possible problem, as chord weights across the piece change and influence each other, making the calculation of chord labels much more difficult.

Passages where the error class ''passing tone called seventh by program'' arise would probably coincide with disagreement among human analysts. This problem class might perhaps be resolved through a deep understanding of structural voice leading.

This completes our description of the labeling system, its performance, error classes, and directions for future improvements. We now turn to the issue of generating a good segmentation of the music.

Computational Issues in Segmentation

Segmentation can be a difficult problem, as we describe in this section. The nature of the problem is important to understand, as it guided the design of the algorithms we describe in this article and affects all algorithms that perform segmentation.

We begin with the simplest case: block chords. With block chords, where notes start and end simultaneously, the harmonic changes are obvious, and so harmonic analysis is relatively straightforward. Music, however, is not always this simple: chords are often arpeggiated, incompletely stated, and interspersed with non-harmonic tones (often stating the melody). All of these things confound the analysis task.

Harmonic change can only occur where notes begin or end. A point of possible harmonic change is called a partition point, where the set of pitches currently sounding in the music changes by the onset or offset of one or more notes. We refer to the ordered set (start to end) of all partition points for a piece of music as P_{all}. The size of P_{all} is denoted as p. A segment is the interval between partition points. A minimal segment is the interval between two sequential partition points. Figure 2 illustrates these concepts.

A segmentation of a piece is a subset of P_{all} containing at a minimum the first and last elements of P_{all}. A binary number uniquely labels any segmentation of a piece of music. The ith bit of the number is 1 if the ith partition point in P_{all} is in the segmentation; otherwise, the bit is 0. The number of partition points in Figure 2, denoted by p, is six. The figure shows a segmentation of the music into segments labeled ''C major,'' ''G7,'' and ''C major,'' respectively. The six-digit number 100111 represents this segmentation.

Because a segmentation is uniquely labeled with a binary number of length p, and the initial and final bits of this number are always 1, there are 2^{p-2} ways to segment any piece of music. The number of partition points is limited to a maximum of twice the number of notes (one point for each note beginning and each note end). Thus, a piece with notes that are staggered so that each note onset and offset is different from any other note onset or offset will have $2^{p-2} = 2^{2n-2}$ segmentations. A single block chord, where all notes begin and end simultaneously will have $2^{2-2} = 2^0 = 1$ segmentations.

Although the number of partition points varies greatly from piece to piece, even a short piece of typical piano music will have hundreds of partition points and thus a huge number of segmentations. Bach's *Sinfonia No. 1*, a 21-measure solo keyboard piece, has 341 partition points and thus 2^{339}, or roughly 10^{102}, segmentations.

Clearly, an exhaustive exploration of possible segmentations is impractical for all but the smallest pieces. Thus, any system that performs harmonic analysis must make assumptions to reduce the size of the search problem for a typical piece of music. In particular, the assumption that the task of labeling a single segment has limited context sensitivity greatly reduces the complexity of the task by allowing the reuse of sub-solutions between segmentations. In other words, if one can assume that the analysis of the first measure of a piece is not affected by the analysis of the 437th measure, comparing two segmentations that differ only in the 437th measure will not require two separate analyses of the first measure (and every measure between the first and 437th). This greatly reduces the overall amount of work. The framework we propose in this article limits context sensitivity in order to constrain the search problem.

Dynamic programming is a search implementation approach designed to reuse sub-solutions. Temperley and Sleator (1999) describe a search based on this approach for finding the best sequence of root labels of a tonal composition. Labels for minimal segments are given numeric scores in a context-independent manner (allowing reuse of sub-solutions). A start-to-finish search for the best path through the labels is performed by adding the score of the best path through the previous segment with the score of the current segment's labels. Label changes between segments receive a numeric penalty. When the end of the piece is reached, the highest-path score corresponds to an optimal labeling, given their scoring measure.

The approach of Temperley and Sleator can be described as a way to solve the optimization problem of finding a highest-scoring labeling of the chord roots for a piece of music. This approach served as an inspiration for our own work, and we, too, have framed the problem as one of numeric optimization and search for the highest-scoring labeling for a piece of music. The approach we have taken, however, has lead to a different formalization of the problem: that of segmentation as graph search. This formalization has lead to two distinct algorithms for solving the problem (one describable as dynamic programming), detailed in the following sections.

Segmentation as Graph Search

Finding the best segmentation requires a metric for assessing the ''goodness'' of a segmentation that allows direct comparison with other segmentations. We do this by generating scores for the individual segments constituting a segmentation. The score of a segmentation is given by the sum of the scores of its component segments. Segmentations can then be directly compared and the best one selected.

For the purpose of this analysis, we assume that individual segments may be scored without reference to their context. The segment-labeling method described in this article has this property.

Figure 9. Creating a Directed, Acyclic Graph (DAG) for the Segmentation Problem.

1. Create one vertex for each partition point.
2. Sort the vertices according to partition-point number.
3. Make an edge between each vertex, i and every other vertex j, where i<j.
4. Make the reward (cost) for each edge i,j equal to the score of the segment from partition point i to partition point j.

Table 3 shows the best-scoring label for each segment in Figure 2, along with the label's score. As the table shows, segments containing two or more chords tend to get low scores. Segments containing notes belonging to a single chord tend to get high scores. The score for an individual segment should correspond to how much sense it makes to select that segment as a unit of analysis for finding a chord label. Thus, the score of the high-scoring label for a segment serves as a reasonable score for that segment. For the remainder of this article, we assume the use of segment scores based on best label scores.

Once a segment is scored in the course of labeling a segmentation, its label and score may be reused in the course of labeling another segmentation. The search algorithm can then build the highest-scoring segmentation out of individually scored segments.

Assuming scored segments, we can pose the segmentation problem as a search for the highest-reward path through a directed acyclic graph (DAG). Two advantages of using this representation are that the search method is intuitive and there are known optimal algorithms we can use. Figure 9 describes the steps taken to create the DAG.

In this DAG, each vertex represents a partition point, and each edge represents one segment. The source vertex represents the initial partition point, and the terminal vertex is the final partition point. Any path from the first vertex to the last one represents a segmentation of the piece. The problem then becomes one of finding the highest-scoring path from the start to the end vertex.

Figure 10 shows a short passage of music represented as a DAG. Partition points (vertices) are labeled with small digits. Two different segmentations are represented on the graph. The path along the gray edges represents the segmentation whose scores are shown in gray. The path along the black edges corresponds to the segmentation scored in black. Large digits of the appropriate color indicate segment scores.

Segmentation Using the Relaxation Algorithm

Posing the segmentation problem as graph search lets us take advantage of the Relaxation algorithm (Cormen et al. 1990), which is guaranteed to find the best path through a topologically sorted DAG in $O(E)$ steps, where E is the number of edges. Because the creation of a DAG from a piece of music inherently sorts the vertices, we can use this algorithm to find the best segmentation. Figure 11 outlines a search algorithm based on the Relaxation algorithm.

From Figure 10, it is clear that the ith vertex (partition point) has i–1 edges (segments) connecting it to the vertices before it. Adding the incoming edges, a piece with p partition points has $0 + 1 + 2 + 3 + \ldots p - 1 < p^2/2$ edges (segments). This can also be seen from the size of Table 3.

The number of partition points is limited to twice the number of notes (one point for each note beginning and each note end). This means a piece with n notes will never have a number of segments exceeding n^2. Because the number of segments equals the number of edges in the DAG, this algorithm finds the best segmentation (assuming a fixed-time segment scoring function) in $O(n^2)$ steps.

Recall that Bach's *Sinfonia No. 1* has 341 partition points, and thus $2^{(341-2)} = 2^{339}$, or roughly 10^{102}

Figure 10. Segmentation as a Directed Acyclic Graph (DAG).

Figure 11. Relaxation-based search algorithm.

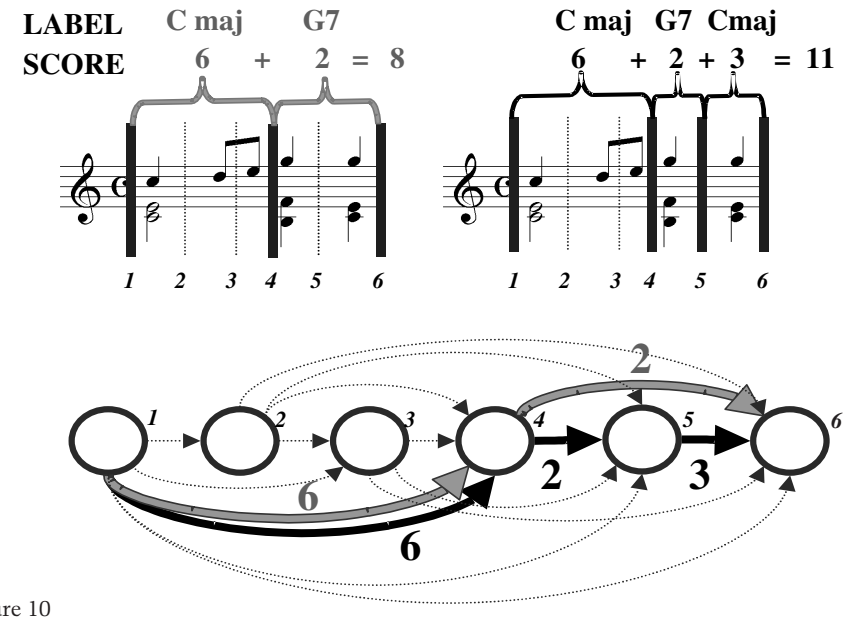


Figure 10

```
1.  for each vertex v, taken in temporal order
2.          for each vertex u having an edge terminating at v, taken in temporal order
3.                  if (score(u) + score(edge(u,v)) > score(v)
4.                          predecessor(v) = u
5.                          score(v) = score(u) + score(edge(u,v))
```

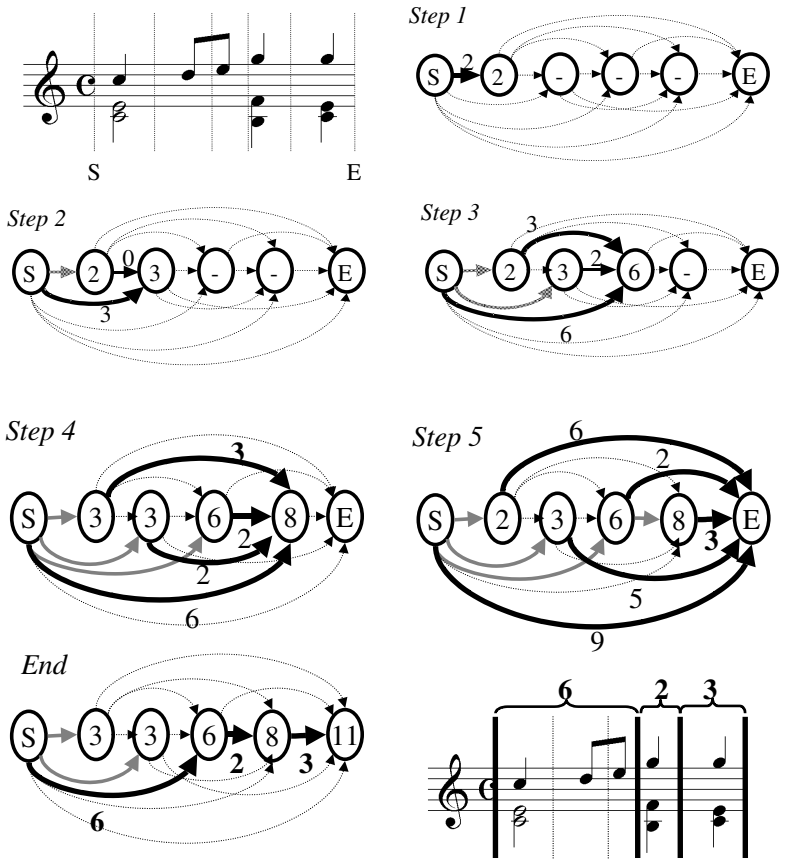
Figure 11

segmentations. Using a Relaxation-based search with context-free, fixed-time segment labeling, the highest-scoring segmentation may be found in $(341^2)/2$ or roughly 58,000 steps.

Figure 12 illustrates the use of Relaxation-based search to find the best segmentation of a measure of music. Each step in the figure corresponds to one iteration of the main loop in Figure 11, starting with the second partition point (since the first partition point has no predecessors). In this figure, black lines correspond to the segments being scored in the current step. The number accompanying each segment (edge) is the score of the best-matching chordal label for the segment. Gray lines correspond to the link from a vertex to the predecessor that generated the best overall path score to that vertex. Numbers in each vertex (partition point) correspond to the score of the highest-scoring path (segmentation) found from vertex *S* (the start of the music) up to that point. All segment scores were generated using the segment labeling method described in another section of this article and correspond to those in Table 3.

Knuth and Plass (1981) formulated the problem of breaking a paragraph of words into optimal length lines as a graph-search problem. The speed of their solution depends on the fact that the maximum allowed line length is known in advance. This limits the context to a relatively small number of breakpoints (usually no more than twenty), and it lets one force a break when a line has gone on too long (i.e., off the page). Since the problem restarts roughly every twenty breakpoints, they search a sequence of graphs, each with no more than twenty partition points. This translates into a search where the additional work caused by a new

Figure 12. Segmentation using full search.



breakpoint is bounded by a fixed value (the size of a graph with twenty vertices). Thus, the computational complexity of the search is $O(n)$, where n is the number of breakpoints.

Unfortunately, one does not know the harmonic rhythm of a piece of music beforehand. It may be that the entire piece consists of a single chord or that it breaks into a thousand chords. The rate at which chords change can also vary drastically throughout the music. It is impossible to say that a chord can have a maximal length of, say, twenty segments. Thus, we are forced to do a full search on the graph of the entire piece if we are to find the optimal segmentation. Still, the idea of a fixed maximum search for each new partition point is a useful one that we employ in our real-time algorithm, the HarmAn algorithm.

The HarmAn Segmentation Algorithm

As stated previously, it is possible, given a good fixed-time segment scoring method, to find the best segmentation of the music in $O(n^2)$ steps. While $O(n^2)$ steps is much better than $O(n^2)$, the time required to process a piece of music still increases quadratically as the number of notes increases linearly. This is not a desirable feature if one expects to use the analysis system for a real-time application, such as an interactive jazz accompanist. For this, it would be preferable to have a system that performs analysis in linear time, taking no more than a fixed number of steps per note. To this end, we have created the HarmAn algorithm for finding a good (although not necessarily the

Figure 13. The HarmAn segmentation algorithm.

```
1.  Mark the initial vertex (partition point) S and final vertex E
2.  While there exists at least one umarked vertex in the DAG
3.          Select an unmarked vertex v from the graph
4.          Select u, the vertex whose only path to v has no intermediate vertices
5.          Select w, the vertex whose only path from v has no intermediate vertices
6.          if score(edge(u,w)) < score(edge(u,v)) + score(edge(v,w))
7.                  mark v
8.          else
9.                  remove from the graph v and all edges connected to v
```

best) segmentation in linear time. Figure 13 describes the HarmAn algorithm.

To use this algorithm for chord labeling in a real-time system, the selection of a vertex (step 3 in Figure 13) must be done in temporal order. Also, only the edges (segments) actually considered in the search should be scored and labeled. If real-time performance is not an issue, then vertices may be selected in reverse order or even random order, and the full set of edges may be scored and labeled in advance. Figure 14 shows an example segmentation performed with HarmAn search using the time-ordered selection of partition points.

In Figure 14, each step represents the consideration of a single vertex (partition point). Black edges represent newly scored segments. Gray edges are previously scored segments remaining in the graph. Dashed edges are not scored. Numbers in each vertex (partition point) correspond to the score of the highest-scoring path (segmentation) found from vertex *S* (the start of the music) up to that point. All scores for segments were generated using the segment-labeling method described in this article.

The HarmAn algorithm considers exactly three edges in the course of evaluating each partition point. The number of partition points is at most twice the number of notes (one for each note beginning and each note end). This results in a maximum of six segment-evaluations per note. Thus, a piece with n notes may be analyzed in $O(n)$ steps.

Returning to the example of Bach's *Sinfonia No. 1*, recall that the piece has 341 partition points, resulting in roughly 10^{102} segmentations. A full Relaxation-based search found the highest-scoring segmentation in $(341^2)/2$ or roughly 58,000 steps. HarmAn search, using the same segment-evaluation method, found a path from start to finish (i.e., a segmentation of the piece) in $(341 - 2) \times 3 = 668$ steps. The catch is that a HarmAn search, unlike a Relaxation search, does not explore all possible segmentations. How good, then, are the solutions HarmAn finds?

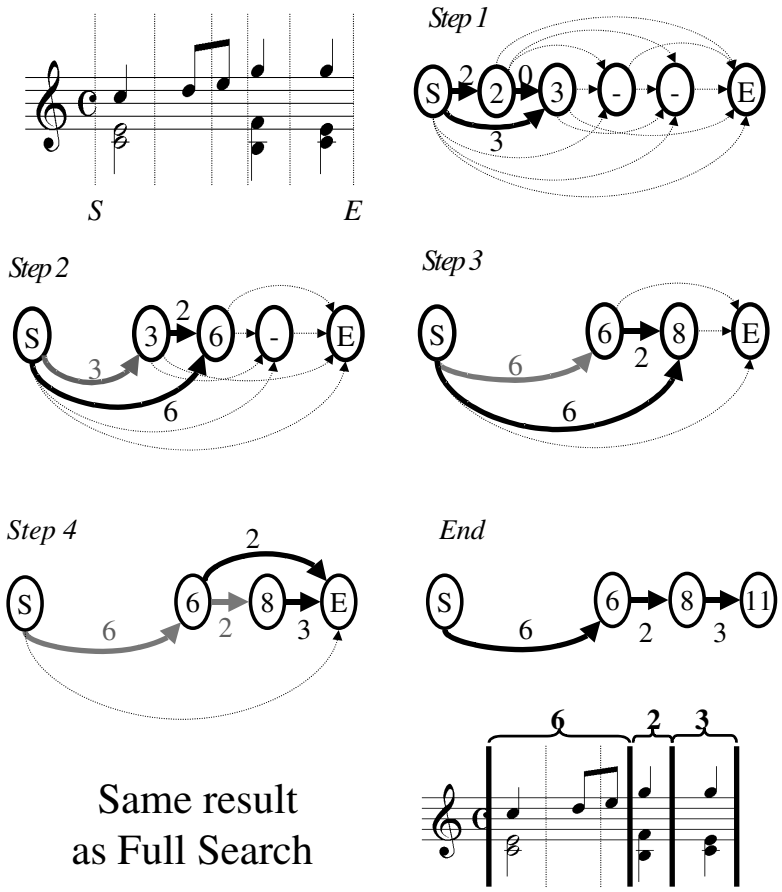
Evaluation of the HarmAn Search Heuristic

A perfect segment-scoring method would assign the highest scores and appropriate labels to the segments of music corresponding to the correct segmentation as determined by some outside measure (such as the harmonic analysis provided for an excerpt in a music theory textbook). We have already shown how a set of partition points and scored segments for a piece of music may be translated into a DAG. To analyze the HarmAn search heuristic, we use our chord-labeling method to score the segments, working under the assumption that this generates perfect segment scores.

Relaxation search is guaranteed to find the highest scoring path through a DAG given a particular metric. The price for this guarantee is that the search considers increasing number of edges (segments) for each new vertex (partition point) encountered in the graph. In the worst case, this results in a running time that is the square of the number of notes in the piece. Because we are interested in developing systems that can run in real-time, HarmAn was conceived as a kind of "greedy" algorithm. That is, it picks the best local, short-term choice at every turn to work on the music in a start-to-finish way, using a fixed maximum number of steps per vertex.

The rationale behind this approach is that music unfolds in time from start to finish. Composers

Figure 14. Segmentation with Time-Ordered HarmAn Search.



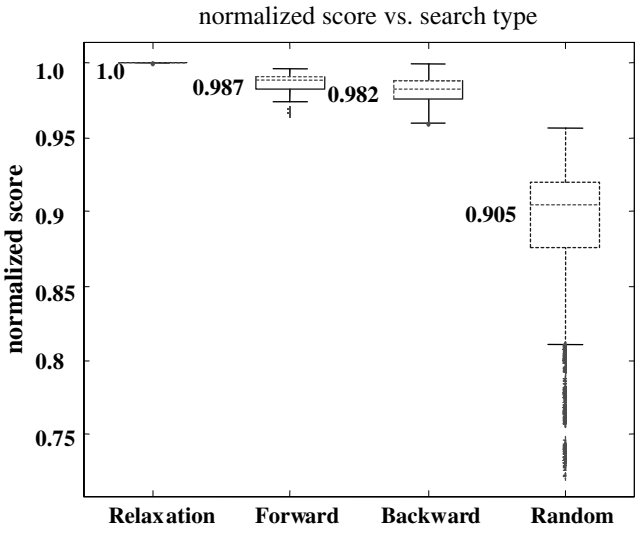
generally create structures that can be understood with limited backtracking to previously heard passages. The expectation for what comes next is determined by what has just been heard. It is reasonable to assume that the set of likely segments under consideration by a human is greatly constrained by what has already transpired in the music. Thus, a ''greedy'' start-to-finish approach is a reasonable way to approximate the first hearing of a piece.

If such an approach is reasonable, and the segment scoring method is a good one, then the following suppositions should also be true. First, segmentations generated by HarmAn search should generate scores very close to those generated by Relaxation search. Second, the order in which HarmAn considers the partition points (start-to-finish, finish-to-start, random, etc.) should strongly affect the resulting segmentation. Third, start-to-finish should be the best HarmAn search order through the partition points, and this should be reflected in higher-scoring segmentations.

In order to test these suppositions, we assembled a corpus of 32 MIDI performances of complete pieces of Western tonal piano music. These were Bach's 15 *Three-Part Inventions*, seven *Bagatelles* from Beethoven's *Opus 33*, and Chopin's *Nocturnes 10* through *19*.

A complete DAG was generated for each piece. The score for each edge in the graph was the score of the best-matching template for that segment, as described in the section on segment labeling.

Figure 15. Normalized segmentation scores.



Each piece was then segmented 43 times. One segmentation was generated by Relaxation search. One segmentation was generated by HarmAn's processing the partition points in start-to-finish order. Another segmentation was generated by HarmAn's processing the partition points in finish-to-start (reverse) order. Finally, 40 segmentations were generated by HarmAn's processing partition points in 40 different random orders. Figure 15 contains a ''box and whisker'' plot for each type of search. All trials for all 32 pieces in the corpus were included in the data for this figure. Thus, for Relaxation, HarmAn forward, and HarmAn backward, $n = 32$. For random, $n = 32 \times 40 = 1{,}260$.

Each box has lines at the lower quartile, median, and upper quartile values. Median values for each search type are also identified by their numeric value. The whiskers are lines extending from each end of the box to show the extent of the rest of the data. ''Outliers'' are data with values beyond the ends of the whiskers. These ''outliers'' are indicated by the plus (''+'') symbol.

Recall that, because Relaxation performs a full search of the segmentation graph, its score is always 100% of the possible score. Note that this may not correspond to generating perfectly labeled segmentations compared to an outside measure. A perfect score in this experiment indicates only that the high-scoring path through a DAG was found. If the segment-scoring algorithm is faulty, then this high-scoring path will not correspond to the best segmentation of a piece of music.

HarmAn search, when applied to the partition points in a start-to-finish (forward) manner, achieves segmentation scores with a median 98.7% of the scores generated by Relaxation search. Finish-to-start (backward) search achieves a median 98.2% of the maximal possible score. Neither forward nor backward search ever achieved a score below 95% of the maximal one. These findings support Supposition 1 and indicate that time-ordered HarmAn search generates good segmentations when compared to full search using Relaxation.

Using random-order selection of partition points for the HarmAn method generates significantly lower scores. Random search had a median score of 90.5% of the maximal score. The lowest score generated by random order selection was on Beethoven's *Bagatelle No. 3*, which received 72.2% of the maximal score. The distinctly lower scores achieved by random-order consideration of partition points support Supposition 2. The order in which the partition points are selected strongly influences the result of the search.

Supposition 3 was not strongly supported by the data. While forward search did achieve the highest median score, it was only marginally better than backward search, and the score distributions for the two orders are almost entirely overlapping. Furthermore, backward search outscored forward search on many pieces.

Evaluation of the Labeling Method for Segment Scoring

This article has explored the performance of the segment-labeling algorithm, assuming a good segmentation has been provided. Two different segmentation algorithms have also been evaluated under the assumption that a good method for scoring segments has been provided. Recall that, in the context of segmentation, a good segment scoring

Figure 16. Grading the output of the combined labeling and segmentation algorithms.

1. Generate minimal segments from a MIDI file representing the piece of music, *M*.
2. Using our segment-labeling method, make a table, *T*, of scored and labeled segments like that in Table 3
3. Generate a Directed Acyclic Graph for the piece, using the segment scores from *T*.
4. Segment one of three ways (Relaxation search, Time-ordered HarmAn, and use of the answer-key segmentation).
5. Apply the tie-breaking rules in Figure 4 to the labeled segments in the resulting segmentation.
6. Grade the labeled segmentation against the answer key as in Figure 5.
7. Repeat steps 4 through 6 until all segmentation methods have been graded.
8. Compare the resulting grades.

method gives high scores to segments that correspond to the ones a human would select as units of harmonic analysis.

We observed that our segment-labeling method could be reasonably used to generate segment scores, because segments containing notes belonging to a single chord tend to get high scores, and those encompassing several chords tend to get low scores. However, the question remained as to how good these segment scores were for finding a segmentation of music that would correspond well with one generated by a human expert. In order to test the quality of scores generated by our segment-labeling approach, we performed the steps in Figure 16 on each excerpt in the KP corpus.

Table 5 shows mean scores for the full corpus of 45 excerpts, comparing labeled segmentations generated with Relaxation search and HarmAn search to the program's labeling of the segmentations from the answer key. The table shows both HarmAn and Relaxation search achieve roughly equivalent performance in the range of 76% correct on the full corpus. This compares to the 88.65% mean score achieved by the labeling system when paired with the segmentation provided by the answer key.

It is instructive to examine the grades achieved by the different search methods on the 13 excerpts that were labeled identically to the answer key when the correct segmentation was provided. Table 5 shows mean scores on these excerpts, broken down by segmentation method. Figure 17 plots the individual grade for each of the 13 excerpts, broken down by segmentation method.

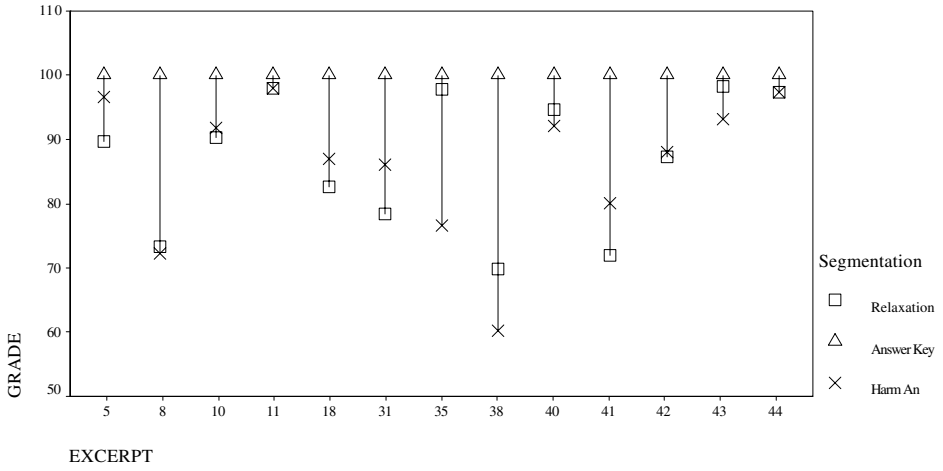
Table 5. Labeling scores by segmentation approach

Proportion of KP Corpus	*Segmentation Approach*	*Mean Grade %*	*Std Dev*
All 45 excerpts	Use Answer Key Segmentation	88.65	10.72
All 45 excerpts	Relaxation Search	76.50	12.88
All 45 excerpts	HarmAn Search	75.81	12.78
Best 13 excerpts	Use Answer Key Segmentation	100.00	0.00
Best 13 excerpts	Relaxation Search	86.89	10.59
Best 13 excerpts	HarmAn Search	86.08	11.13

Table 5 shows that the Relaxation search correctly labeled 87% of minimal segments on the 13 excerpts. Because the system achieves perfect labeling on these excerpts when the correct segmentation is provided, the search methods must have generated segmentations different from those given by the answer key. Relaxation search is guaranteed to find the best scoring segmentation given a particular scoring metric. Thus, the only way for the search not to find the same segmentation as the answer key is for the segment scoring mechanism to be sub-optimal for the task of finding the right segmentation.

This conclusion is reinforced by the fact that HarmAn search performs better than the Relaxation search on several cases. The difference between the scores indicates that HarmAn search

Figure 17. Segmentation grades on the 13 best excerpts.



does not generate identical segmentations to the Relaxation search. Since the relaxation search finds the highest scoring segmentation given the internal segment scoring metric, HarmAn search can only beat the Relaxation search if the internal scoring metric does not always capture all the information needed to segment the passage correctly.

Of course, raw grades are meaningless without context. An analysis system that achieves a mean grade of 87% is far from perfect, but it may be good enough for many tasks. Consider Figure 18, which shows excerpt 11, and Figure 19, which shows excerpt 41. The segmentation methods both scored very well (98%) on excerpt 11 and rather poorly (72% and 80%) on excerpt 41. Examination of the analyses of these two excerpts helps illustrate what the numeric grades mean.

In both figures, the answer key is shown at the top of each system. The program's output is shown on the bottom. Chord labels apply from the point where they appear until the next label appears. Discrepancies between the answer key and program output are identified by dashed rectangles indicating the duration of the discrepancy.

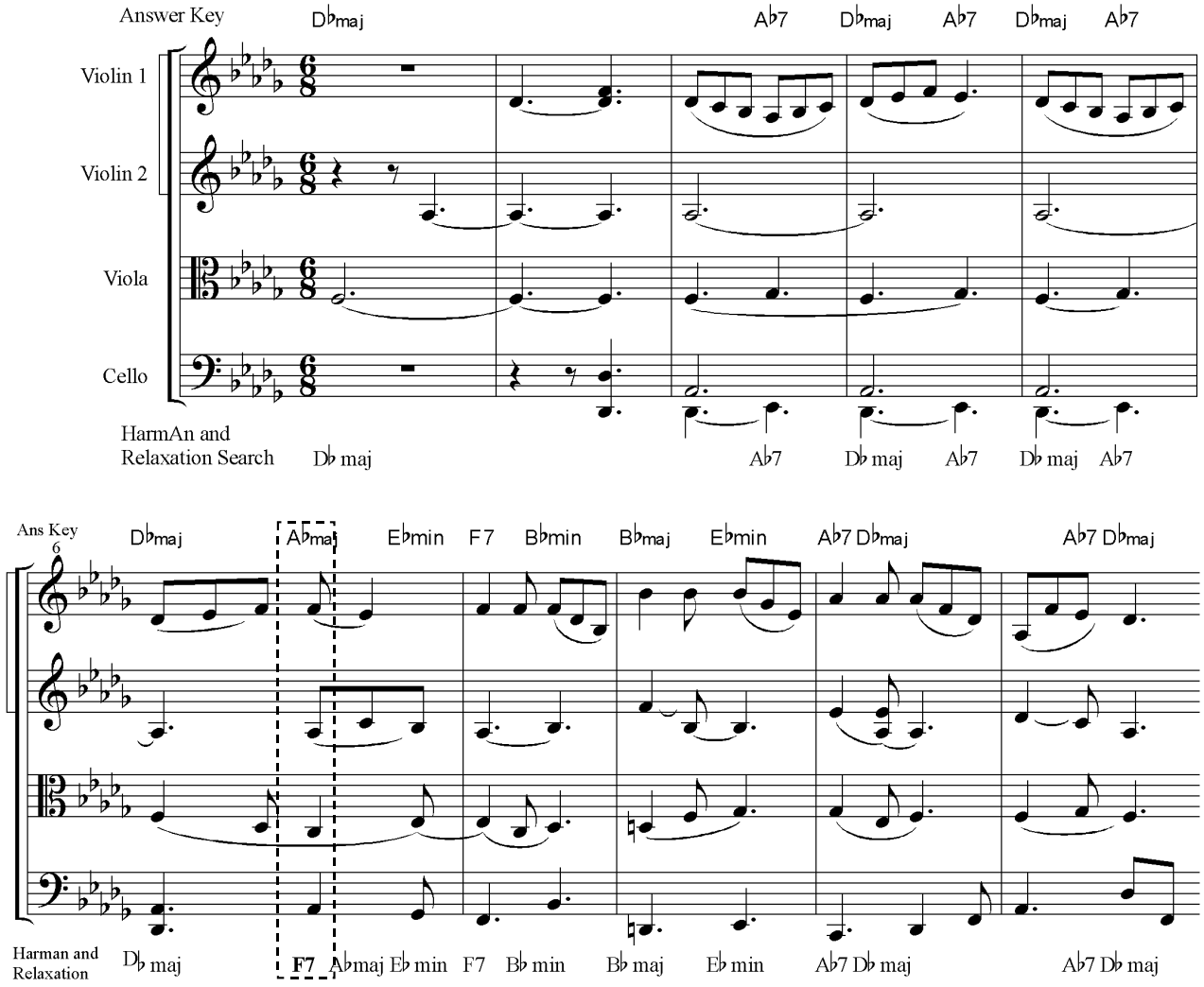
Excerpt 11 (Figure 18) was drawn from the third movement of Beethoven's *String Quartet Op. 135* and received a score of 98% when segmented using either HarmAn or Relaxation search. Here, the two searches generated the same segmentation and labeling. Measure six contains the only error on this excerpt and is the result of a passing F7 chord not acknowledged as part of the harmony in the answer key.

Excerpt 41 (Figure 19) shows measures five through 17 of Schumann's "*Aus meinen Thränen spriessen*." Here, HarmAn search achieved a score of 80%, while Relaxation search scored 72%. The labeling shown in this figure is that of the lower-scoring Relaxation search. The two searches differ in the treatment of the B at the very end of measure 11. HarmAn, when searching in temporal order, tends to continue the current segment a long as possible and thus assigns the B to the G-sharp half-diminished chord in measure 11. The following measure is then correctly labeled C-sharp major. Relaxation search unifies the B at the end of measure 11 with the C-sharp triad in the following measure, resulting in a label of C-sharp 7, which proves to be incorrect.

The remainder of the analysis is identical between the two segmentation methods. Measure one shows a situation where the initial A major triad was unified with the F-sharp minor following it. This happens because only the root and third of the A major are present. Because both of these notes are also part of the F-sharp minor triad, the program does not assign separate labels to beats one and two.

Measures 7, 8, 15, and 16 all show the same error. Here, the human analyst chooses to begin the

Figure 18. Excerpt 11, Beethoven's String Quartet Op. 135, III, mm. 1–10.



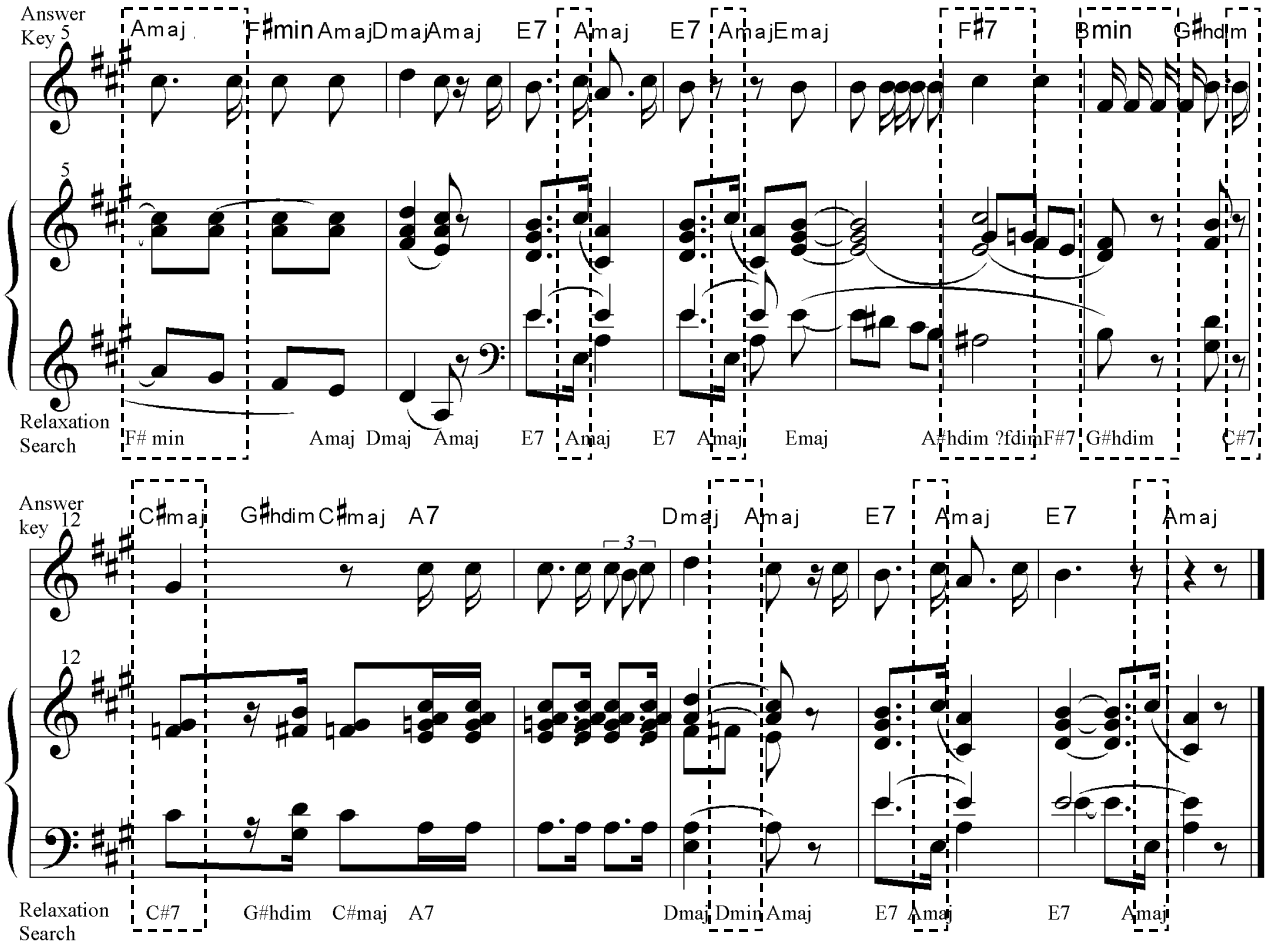
A minor chord on the second beat of the measure, while the program (noting the C on the second half of beat one) begins the A minor triad on the second half of the first beat. Presumably, the human favors the second beat owing to the appearance of the root of the A minor chord.

The errors in measures 10 and 14 are both the result of spurious vertical simultaneities that happen to spell chords. Because the segment scoring metric is based on how well a particular segment matches a particular chord label, both search methods tend to select segments that exactly spell a chord, even if the chord is clearly (to a human) the result of passing tones.

Conclusion

In this article, we described a system for segmenting and labeling tonal music based on template matching and graph-search techniques. We then analyzed both the computational complexity of the

Figure 19. Excerpt 41, Schumann, "Aus meinen Thränen spriessen," mm. 5–17.



algorithms used and real-world system performance by comparing the analyses it produced to those created by human analysts. The results produced by this system are surprisingly good, given its simplicity, and provide insight into the computational problems in analysis of tonal music.

When provided the correct segmentation, the segment-labeling method described in Figures 3 and 4 labeled an average of 89% of minimal segments correctly on the KP corpus of musical excerpts. Thirteen out of 45 excerpts in this corpus were labeled perfectly using this method.

Our analysis of the errors generated by the labeling system suggests further improvements to the system. Figure 8 shows error class frequencies on the corpus and indicates, for example, that 26% of labeling errors can be eliminated with improved tie breaking between chord templates.

The combined segmentation and labeling system uses either Relaxation or HarmAn search to find a good segmentation based on segment scores generated by the labeling system. This system correctly labels roughly 77% of minimal segments in the full KP corpus. These results are surprisingly good, given the unforgiving nature of the grading scheme and the simple analytical approach used by the system. Recall that metric information, key signature, harmonic function, voice leading, and stylistic in-

formation were not used. Furthermore, the assumption that segments can be labeled in isolation, without regard to context, is a strong one that, intuitively, one would expect to have far more of a negative impact than turned out to be the case.

The evidence supports the conclusion that the search methods we are using are good. Given our formulation of the segmentation problem as a search for the best path through a Directed Acyclic Graph, it is unlikely that a computationally more efficient algorithm than Relaxation can be found without giving up either accuracy or completeness of the search result. HarmAn search is an example that gives up completeness to gain a time advantage over Relaxation search, while still returning nearly equivalent results.

While we broke the chordal analysis problem into two quasi-independent subtasks, it is clear that there are subtle (and not so subtle) interactions when segment scores are generated using the segment labeling system. Furthermore, segment scores generated by our current system do not fully capture the detail needed to match the answer key segmentation. This is shown most clearly by the results on the thirteen excerpts the system labels perfectly when given the correct segmentation. Table 5 shows that Relaxation search correctly labeled 87% of minimal segments on these thirteen excerpts. Because the system achieves perfect labeling when the correct segmentation is provided, Relaxation must have generated incorrect segmentations. Relaxation search is guaranteed to find the best scoring segmentation, given some scoring metric. Thus, the only way for the search not to find the same segmentation as the answer key is for the current segment scoring mechanism to be suboptimal for the task of finding the right segmentation.

Improving the segment scoring is a difficult task, and one that we plan to investigate further. Clearly, the scheme we use to assign positive and negative evidence is not sufficient, but improving it while maintaining generality is not easy. This leads us to consider segment scoring mechanisms that are either directed to a particular style (e.g., a ''Kostka-Payne'' mechanism that favors what their textbook favors), or that use additional information, such as metric placement of notes and dynamic contrasts.

The system and experiments in this article contribute a number of things to the field of musical analysis by computer. We describe our approach to a level of detail that allows others to duplicate and extend the work. This article examines the computational complexity of the algorithms we use to solve the task. Finally, the performance of the system is measured using a clear grading measure against an objective standard on a corpus of known pieces, providing an empirical baseline against which future work may be measured.

Acknowledgments

We would like to thank David Temperely for providing the Kostka-Payne corpus and for many helpful discussions. We also thank Roger Dannenberg for his helpful comments. We gratefully acknowledge the support of the NSF under award IIS-0085945 and a University of Michigan College of Engineering Seed Grant. The views expressed in this article are solely those of the authors and do not necessarily reflect the views of the funding agencies.

References

Cormen, T., et al. 1990. *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press.

Ebcioglu, K. 1992. ''An Expert System for Harmonizing Chorales in the Style of J. S. Bach.'' In M. Balaban, K. Ebcioglu, and O. Laske, eds. *Understanding Music With AI: Perspectives on Music Cognition.* Cambridge, Massachusetts: MIT Press, pp. 294–333.

Forte, A. 1973. *The Structure of Atonal Music*. New Haven, Connecticut: Yale University Press.

Knuth, D., and M. Plass. 1981. ''Breaking Paragraphs into Lines.'' *Software Practice and Experience* 11:1119–1184.

Kostka, S., and D. Payne 1984. *Tonal Harmony*. New York: McGraw-Hill.

Laden, B., and D. H. Keefe. 1989. ''The Representation of Pitch in a Neural Net Model of Chord Classification.'' *Computer Music Journal* 13(4):12–26.
Maxwell, J. H. 1992. ''An Expert System for Harmonizing Analysis of Tonal Music.'' In M. Balaban, K. Ebcioglu and O. Laske, eds. *Understanding Music With AI:Perspectives on Music Cognition.* Cambridge, Massachusetts: MIT Press, pp. 335–353.
Pardo, B., and W. P. Birmingham. 2001. ''The Chordal Analysis of Tonal Music.'' Technical Report CSE-TR-439–01. Ann Arbor, Michigan: Department of Electrical Engineering and Computer Science, The University of Michigan.
Smaill, A., et al. 1993. ''Hierarchical Music Representation for Composition and Analysis.'' *Computers and the Humanities* 27:7–17.
Smoliar, S. 1980. ''A Computer Aid for Schenkerian Analysis.'' *Computer Music Journal* 4(2):41–59.
Taube, H. 1999. ''Automatic Tonal Analysis: Toward the Implementation of a Music Theory Workbench.'' *Computer Music Journal* 23(4):18–32.
Temperley, D., and D. Sleator. 1999. ''Modeling Meter and Harmony: A Preference-Rule Approach.'' *Computer Music Journal* 23(1):10–27.
Ulrich, J. W. 1977. ''The Analysis and Synthesis of Jazz by Computer.'' In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*. Los Altos, California: William Kaufmann.
Wakefield, G. H. 1999. ''Mathematical Representation of Joint Time-Chroma Distributions.'' Paper presented at the International Symposium on Optical Science, Engineering, and Instrumentation, SPIE'99, 18–23 July, Denver, Colorado.
Widmer, G. 1992. ''Perception Modeling and Intelligent Musical Learning.'' *Computer Music Journal* 16(2):51–68.
Winograd, T. 1968. ''Linguistics and the Computer Analysis of Tonal Harmony.'' *The Journal of Music Theory* 12:2–49.

Appendix: The KP Corpus

The excerpt number, composer, and composition are shown for each member of the KP corpus below.

1. Anonymous (often attributed to Bach), *Minuet in G, mm. 1–16*
2. Bach, *Chorale, ''Uns ist ein Kindlein heut' geborn''*
3. Bach, *Chorale, ''Jesu, der du meine Seele''*
4. Beethoven, *Rondo Op. 51, no. 1, mm. 103–120*
5. Beethoven, *Sonata Op. 10 No. 1, II, mm. 1–8*
6. Beethoven, *Sonata Op. 10 No. 3, II, mm. 9–17*
7. Beethoven, *Sonata Op. 13, II, mm. 1–8*
8. Beethoven, *Sonata Op. 14 No. 1, III*
9. Beethoven, *Sonata Op. 2 No. 3, III, mm. 81–88*
10. Beethoven, *Sonata Op. 27 No. I, mm. 1–9*
11. Beethoven, *String Quartet Op. 135, III, mm. 1–10*
12. Beethoven, *String Trio Op. 9 No. 3, II, mm. 1–10*
13. Brahms, *''Und gehst du ueber den Kirchhof,'' Op. 44, mm. 29–37*
14. Ayer (arr. Campbell), *''Oh! You Beautiful Doll!,'' mm. 1–9*
15. Chopin, *Mazurka Op. 62, No. 7, mm. 1–16*
16. Chopin, *Mazurka Op. 63, No. 2, mm. 1–16*
17. Chopin, *Nocturne Op. 27 No. 1, mm. 41–52*
18. Grieg, *''The Mountain Maid,'' Op. 67 No. 2, mm. 1–11*
19. Haydn, *Sonata No. 22, III, mm. 1–8*
20. Haydn, *Sonata No. 30, I, mm. 84–96*
21. Haydn, *String Quartet Op. 20 No. 4, I, mm. 13–24*
22. Haydn, *String Quartet Op. 50 No. 6, II, mm. 55–63*
23. Haydn, *String Quartet Op. 74 No. 3, II, mm. 30–7*
24. Haydn, *String Quartet Op. 76 No. 6, II, mm. 31–9*
25. Mozart, *Bassoon Concerto K. 191, II, mm. 42–50*
26. Mozart, *''Eine Kleine Nachtmusik,'' K. 525, II, mm. 1–8*
27. Mozart, *Piano Concerto K. 488, II, mm. 1–12*
28. Mozart, *Sonata K. 330, II, mm. 21–8*
29. Mozart, *Sonata K. 333, III, mm. 91–8*
30. Mozart, *Piano Trio K. 542, I, mm. 210–229*

31. Mozart, *Marriage of Figaro, ''Voi che sapete,'' mm. 41–52*
32. Schubert, *Sonata in B-flat, D. 960, I, mm. 149–68*
33. Schubert, *''Erlkönig,'' mm. 113–23*
34. Schubert, *''Erlkönig,'' mm. 134–48*
35. Schubert, *''Auf dem Flusse,'' mm. 14–21*
36. Schubert, *Impromptu Op. 90 No. 1, mm. 42–55*
37. Schubert, *String Trio D. 471, mm. 187–201*
38. Schubert, *Originaltanze Op. 9 No. 14, mm. 1–24*
39. Schumann, *''Die beiden Grenadiere,'' mm. 23–37*
40. Schumann, *''Sehnsucht,'' mm. 2–11*
41. Schumann, *''Aus meinen Thränen spriessen,'' mm. 1–17*
42. Schumann, *''Wenn ich in deine Augen seh','' mm. 1–21*
43. Tchaikovsky, *''Morning Prayer,'' mm. 1–17*
44. Tchaikovsky, *''The Nurse's Tail,'' mm. 5–15*
45. Tchaikovsky, *Symphony No. 6, I, mm. 89–97*