Melodic segmentation: evaluating the performance of algorithms and musical experts

Karin Höthker[*], Belinda Thom[†], Christian Spevak[*] [‡]
Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe
bthom@cs.cmu.edu, {spevak, hoethker}@ira.uka.de

Abstract

We review several segmentation algorithms, qualitatively highlighting their strengths and weaknesses. We also provide a detailed quantitative evaluation of two existing approaches, Temperley's Grouper and Cambouropoulos' Local Boundary Detection Model. In order to facilitate the comparison of an algorithm's performance with human behavior, we compiled a corpus of melodic excerpts in different musical styles and collected individual segmentations from 19 musicians. We then empirically assessed the algorithms' performance by observing how well they can predict both the musicians' segmentations and data taken from the Essen folk song collection.

1 Introduction

Segmenting or parsing a melody means imposing a temporal structure on a sequence of discrete pitch symbols. This structure is defined by pairs of boundaries that break the sequence up into various subsequences. It may involve different levels of hierarchy (Lerdahl and Jackendoff 1983), overlapping boundaries (Crawford et al. 1998), and unclassified areas not belonging to any segment. From a computational point of view it makes sense to simplify the task, to focus on breaking a melody into a *segment stream*, which we define as a series of non-overlapping, contiguous fragments. In addition, rather than explicitly dealing with hierarchy, we consider segment streams on two different levels of granularity, the *phrase level* and the *sub-phrase level*. Phrases capture large scale structure of a piece, whereas sub-phrases are more fine-grained, providing additional structure at "the next level down."

An algorithm that automatically produces a musically reasonable segment stream provides an invaluable tool for melodic modeling. The higher-level structure emerging from the melodic surface can be used as an input for further investigations which is what motivates our interest in the subject. For example, Höthker is exploring what higher-level features are required to adequately represent structure in melodies within and across different musical styles. She needs a general-purpose segmentation algorithm in order to consider features that can extend beyond fixed-size contexts. Thom also intends to integrate such an algorithm into *BoB* (Thom 2001) – a real-time agent designed to trade customized solos with an improvising musician – so that she may investigate whether or not the agent's interaction with a user will noticeably improve when its melodic representation is no longer limited to a per-bar representation.

Melodic segmentation and representation are inherently intertwined. What appears salient in a melodic surface will depend on a sequence's encoding – e.g. is the focus rhythmic, intervallic? – and on the way a melody is locally divided when building higher-level features. Ultimately, the development of a melody's encoding, its segmentation, and its higher-level abstractions should be combined into a single algorithm, which could explicitly consider the cooperation and conflict between these different components. However, in the meantime, as such an algorithm has yet to be invented, we strive to understand the components in isolation. For fixed-size melody segments, a detailed investigation of clustering algorithms and melodic encodings is already available (Höthker et al. 2001). This paper is intended to accompany the other, focusing on general-purpose segmentation algorithms.

In Section 2, we begin by reviewing some segmentation algorithms, *qualitatively* highlighting their strengths and weaknesses. The remaining sections focus on *quantitative* evaluation, exploring the behavior of Cambouropoulos' *Local Boundary Detection Model (LBDM)* and Temperley's *Grouper* in a variety of musical settings. Note that our ultimate goal is not to determine which algorithm is better but to obtain a feel for how these algorithms might perform when used off-the-shelf in our applications. As described in Section 4, we ran four quantitative experiments (Exp A through D). The results are interpreted in Section 5.

One reason that quantitative evaluation is challenging is that the segmentation task is *ambiguous* – there might be several or even many musically plausible output segmentations for a given input melody. This situation arises because many different factors influence the perception of salient melodic chunks: local structure (e.g. gestalt principles), higher-level structure (e.g. re-

[*] supported by the Klaus Tschira Foundation
[†] supported by the German-American Fulbright Commission
[‡] Author ordering was determined by stochastic simulation.

curring motives, harmony), style-dependent norms and the breaking of these norms, etc.

If a meaningful quantitative assessment of an algorithm's performance is to be made, melodic ambiguity should be considered. As an important precursor to quantitative investigation, therefore, we had 19 musicians hand-segment melodic excerpts in different musical styles (Section 3). This data is referred to as the *musician corpus*. We quantify how ambiguous a particular excerpt is computing the average dissimilarity within the musicians' solutions (Exp A). This measure provides an upper-bound on how well a computer algorithm can be expected to perform.

LBDM and Grouper both have adjustable parameters that allow a user application to control, for example, to which extent rests or large intervals influence the determination of segment boundaries. An obvious practical question is how to set these parameters, which we explore along two dimensions. *Flexibility* considers how well an algorithm could perform in *principle*, which depends upon how rich its underlying hypothesis space is and how well this can capture the segmentation processes that operate in realistic musical situations. An algorithm's *stability*, on the other hand, considers how practical an algorithm *might* be, evaluating to what extent the performance obtained in one musical domain (a particular style) or circumstance (a particular song) depends upon a specific parameter setting. There is no guarantee that a more flexible algorithm will necessarily be more stable. Musically plausible behavior in one situation will only perform well in another provided that, in spite of fixed parameters, the algorithm can still make the necessary distinctions when segmenting new melodies. We investigate flexibility by observing how well an algorithm can do when its parameters are tuned to two kinds of domain data – large, relatively homogeneous subsets of the Essen folk song collection (Exp D) and the ambiguous data in our musician corpus (Exp B). Stability (Exp C) is measured by observing how well an algorithm can do when its parameters are tuned to a large Essen subset even though its performance evaluation takes place over the data in the musician corpus.

2 Segmentation Algorithms

Various segmentation algorithms have been proposed: rule- vs. memory-based, lower- vs. higher-level, real- vs. non-real-time, etc. In this section, we review several different types, considering both their strengths and weaknesses.

An algorithm's input typically includes a melodic line, encoded as a note list, where each note is defined by a discrete pitch and an onset and offset time. In this paper, we use the following notation: potential boundary locations are indexed by the second index of a successive note pair. As such, an algorithm's output – encoded by a *segmentation vector*, $\mathbf{s} = (\mathbf{s_1}, ..., \mathbf{s_i}, ..., \mathbf{s_n}) \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{n}}$ – contains a boundary between notes $i-1$ and i if and only if $s_i = 1$. An algorithm might also extract higher-level features from its input. For instance, a list of inter-onset intervals *(IOIs)* measures the duration between onsets of successive pitches; a list of offset-to-onset intervals *(OOIs)* measures the rests between successive note pairs; and a list of absolute pitch intervals *(APIs)* measures the *absolute* difference between adjacent pitches in semitones regardless of the interval's direction.

2.1 Grouper

Grouper, a part of the Melisma Music Analyzer,[1] was designed to extract melodic phrase structure from a monophonic piece of music. A fundamental assumption in Melisma is that intelligent music analysis can be computationally simulated using a purely rule-based approach. Grouper is based on a set of phrase structure preference rules adapted from Lerdahl and Jackendoff (1983). A *Gap Rule* prefers to insert boundaries at large IOIs and OOIs, a *Phrase Length Rule* prefers phrases to have a predefined number of notes, and a *Metrical Parallelism Rule* prefers to begin successive groups at parallel points in the metric structure. The first rule depends on the note list, and the second depends upon the structure of a given solution. The last rule, in contrast, requires an additional input, a *beat list*, which specifies the melody's metric structure. Because duration values are normalized, tempo is not used. Grouper also ignores a melody's pitch content.

Grouper begins by calculating a gap score for each pair of notes, summing their IOI and OOI values. Next, a logarithmically scaled penalty is assigned to all boundaries that deviate from some ideal length. Another penalty is assigned to successive phrases that start at different metrical positions.[2] The optimal segmentation is the solution with the lowest sum of penalties. The penalties are controlled by five user-adjustable parameters: a gap weight, an ideal phrase length, a length penalty, and two metric penalties. With 2^n possible segmentations, dynamic programming is used to efficiently calculate an optimal solution.

2.2 LBDM

LBDM is motivated by the gestalt principles of similarity and proximity. It assigns a boundary strength to each pair of notes in a melody, quantifying how discontinuous each note pair is. Peaks in the sequence of strengths suggest where boundaries should be inserted. Cambouropoulos (2001) points out that LBDM is not a complete model of grouping in itself because issues like musical parallelism (Cambouropoulos 1998) are not explicitly handled. Nonetheless, we consider the model as is because of its elegant simplicity, which

[1] Ready-to-use software is available at `www.links.cs.cmu.edu/music-analysis`.

[2] Melisma provides a module for inferring a beat list from a note list. To ensure that this input is error free in our experiments, we generate it directly using a song's meter.

should make it straightforward to apply in real-time MIDI environments. As a result of the model's parsimony, it was also fairly easy to implement.[3]

LBDM begins by transforming its note list into three interval profiles: APIs, IOIs, and OOIs. Per profile, two rules control how the discontinuity associated with each interval is calculated. A *Change Rule* assigns to each interval a value that is proportional to the rate of change between successive interval pairs. A *Proximity Rule* further scales each value proportionally to the size of the corresponding interval. Before combining the profiles into an overall weighted sum, each is normalized over its length. Boundaries are then inserted between those notes whose overall profile values correspond to local maxima. Six user-adjustable parameters include: three profile weights, two offset values,[4] and a threshold parameter. The threshold, which controls how large a maximum must be before it is considered as a potential boundary, has great impact on LBDM's output. It was not specified how to determine the threshold. We chose to define it as a real-valued fraction of the profile's global maximum. Only the highest peak in any region that crossed this cutoff was assigned a boundary. Other schemes, such as returning a fixed number of peaks, are also worth considering.

In the context of content-based musical information retrieval, a segmentation algorithm based on LBDM has been developed as a pre-processing tool for indexing melodies in the music data base SMILE. The algorithm has been tested on a corpus of 20 hand-segmented melodies from four composers and is reported to compute segmentations which are reasonably similar to the musicians' solutions (Melucci, Orio, and Gambalunga 2000).

2.3 Other Approaches

Bod argues for a memory-based approach to segmentation and implemented such a system using probabilistic grammar techniques (Bod 2001). His most sophisticated and successful grammar performs data oriented parsing (DOP), wherein probabilistic trees are learned from a large training set of Essen data. Musically, DOP is interesting because it imposes a higher-level structural definition upon the model that is to be learned, biasing it to prefer solutions that contain some ideal number of segments. Trees are learned directly from melodies encoded in the Essen Associative Code *(EsAC)* (Schaffrath 1997). Pitch, in relation to the tonic, and note length features are used. To implement DOP would involve significantly more work than LBDM, and because we did not have access to a complete ready-to-use version, we do not consider it in our empirical experiments.

[3]Our implementation of LBDM is available at `i11www.ira.uka.de/~musik/segmentation`.

[4]These shift the API and OOI profiles above zero and were suggested in the N.B. comment in Figure 1 of Cambouropoulos (2001). IOIs do not need an offset because in realistic settings they will always be larger than zero.

Some algorithms also explicitly consider harmonic structure. One example is the phrase boundary agent used in the interactive music system *Cypher* (Rowe 1993). This agent collects information about discontinuities from a complex set of features (including register, dynamics, duration, harmony and beat) in real-time and calculates a weighted sum for each event. When the sum exceeds a threshold, the respective event is considered the beginning of a new phrase. This threshold can be adjusted dynamically according to the musical context.

Two other relevant algorithms, one a rule-based system and the other based on a multi-layer neural network, were developed for automatic musical punctuation of a score (Friberg et al. 1998). Both systems receive information about pitch, duration, and melodic tension (determined with respect to the underlying harmonies) and both operate on a local context of up to five notes. A primary way in which our research differs from Friberg's approach is that we focus on ambiguity whereas they focused on handling a single musician's interpretation.

We also do not consider any models that rely on harmonic analysis because, for our purposes, it made sense to decouple this analysis from the segmentation process. One practical reason for this decision is that the Essen database does not provide harmonic information. Furthermore, in BoB, harmonic information may not always be available, and in Höthker's research, where a modular, exploratory approach is sought, one could imagine needing segmentation as a preprocessing step for harmonic analysis.

2.4 Discussion

LBDM is a local algorithm in the following sense: its two rules never consider more than four and two adjacent notes, respectively. An attractive aspect of LBDM is that can operate on raw MIDI data, meaning that rhythmic transcription is not required. As a result, only online peak estimation would need to be added before the algorithm could be used in real time. Using raw MIDI data is also advantageous because it provides *performed* IOIs and OOIs, articulation data that we expect to be relevant in many situations. Another advantage is that the algorithm's boundary strength calculations can provide insight into why certain boundaries were chosen over others. Although it would be non-trivial to convert this data into a probabilistic measure (Spevak, Thom, and Höthker 2002), this information might prove valuable when evaluating LBDM's performance in musically ambiguous situations.

The Grouper algorithm explicitly combines a local view, i.e. the current proposed segment, with higher-level metric structure. Grouper is able to handle raw MIDI data using additional tools provided in the Melisma package. To use Grouper in real-time would require a paradigm shift however, since dynamic programming must scan an entire melody before return-

ing an optimal solution. For some parameter settings and melodies, we have found Grouper hard to understand (Spevak, Thom, and Höthker 2002). One cause is that Grouper's combination of local and more long-term knowledge is fairly complex – dynamic programming allows a solution's fitness to efficiently depend upon its most recent previous boundary as well as local metric information and duration content. We would have found it useful if Grouper had output a profile that indicated how relevant each segment was to its overall solution.

DOP also explicitly combines local and higher-level structure by simultaneously considering an arbitrary-length sequence of most recent notes and a preferred total number of phrases. Given a fairly large training set (around 4000 songs), this algorithm can learn to segment similar unseen melodies (Bod 2001). Learning is an asset – it certainly makes DOP more flexible than a hard-wired algorithm – but reliance on a large, hand-segmented training set is costly. In order for such a scheme to be of practical use, one would need to demonstrate that a model trained in one setting (e.g. Essen folk tunes) might also generalize well in other settings (e.g. music analysis, live performance). Another drawback is DOP's reliance on EsAC, a score-based encoding that precludes the use of raw MIDI data. Although raw MIDI could be transcribed, this would preclude the use of potentially relevant articulation data. We believe DOP's greatest asset is its probabilistic basis, which provides a mechanism for explicitly handling ambiguity. For example, with this model, one should be able to estimate how likely or *fit* a solution is. This fitness could then be used to penalize a solution based on the discrepancy between the algorithm's assessment of its fitness and an external measure that quantifies its musical plausibility.

In contrast to Grouper, LBDM considers pitch content but ignores metric content. Since our present need is to use an algorithm off-the-shelf, we resisted modifying either algorithms' underlying representation, even though one might argue that this would provide a better comparison of the *core* concepts. The results in Section 5 for the unmodified algorithms suggest that in practical situations, pitch content might be less relevant for segmentation of melodies than metric information. However, this conjecture needs to be investigated more closely to be validated. From a user's point of view, when choosing among existing algorithms, the time necessary for parameter tuning is a practical consideration. Here, the expected effort is balanced between Grouper (5 parameters) and LBDM (6 parameters). Since the complexity of optimizing parameters increases exponentially in their number, extending LBDM would have seriously affected this balance.

It is worth noting that much inconvenience can be hidden in a user-adjustable parameter, and it is easy to gloss over important free parameters that need to be specified when an algorithm is extended. For example, with one additional parameter (a weight for integrating a profile that indicates where similar motives occur), one could integrate motivic parallelism into LBDM (Cambouropoulos 1998). To implement this change, however, would require a motive clustering algorithm which would introduce its own set of adjustable parameters. A similar situation arises when learning is used. For example, as opposed to the brute-force search we use to tune LBDM's and Grouper's parameters to a specific dataset, one could imagine using an explicit learning procedure like gradient descent. We would like to see Grouper and LBDM extended in this way, but to do so would require the specification of additional meta-parameters for controlling the learning process (when to stop learning, how much change to incorporate per learning iteration, etc.).

When learning is not incorporated, all degrees of freedom are provided by the model's user-adjustable parameters. Learning introduces more freedom, but it still relies on a hard-wired set of meta-parameters, a fixed input representation, and so on. Certainly, some aspects of the segmentation process are hard-wired (e.g. primitive, gestalt-based), while others rely on cultural or stylistic norms (e.g. familiarity of certain a genre, motive, or musical pattern). Given this realization, perhaps the most promising future direction for algorithmic segmentation is to combine the musically more informative features used by the more hard-wired algorithms (e.g. LBDM, Grouper, Cypher) with the more flexible, probabilistic approach taken in DOP. The motivation for this suggestion is that with better features, less data may be needed to configure a model's free parameters.

3 Segmentation Data

The Essen Folk Song Collection. *Essen* is a large corpus of mostly European folk songs initiated by Schaffrath (1997) and maintained by Dahlig.[5] The Essen data is encoded in EsAC format, recording meter and key, and a sequence of discrete pitches and durations for each folk song. Phrase boundary locations for each song are also included. These are typically modeled after a song's text phrases. Essen data has already been used to assess the performance of DOP and Grouper (Bod 2001; Temperley 2001), but not in a uniform setting.

We use the Essen phrase marks as a reference to benchmark the algorithms' performance because it provides an unambiguous, single-layer segmentation. A better understanding of how the segments were obtained is instructive. Dahlig (2002) outlined the Essen segmentation process as follows:

> "When we encode a tune without knowing its text, we do it just intuitively, using musical experience and sense of the structure. The idea is to make the phrase not too short (then it is a motive rather

[5] Approximately 6000 songs are publicly available at `www.esac-data.org`.

Subset	N_s	Notes per song	Notes per phrase
Kinder	208	39.1 ± 20.9	7.0 ± 1.3
Fink	554	58.7 ± 24.4	8.3 ± 1.6
Ballad	869	37.9 ± 14.2	9.2 ± 1.7
Kaszuby	981	45.6 ± 29.4	11.5 ± 2.9
All	2612	45.3 ± 22.6	9.7 ± 2.1

Table 1: Properties of the Essen data subsets (average $\pm$ standard deviation). N_s is the number of songs in a subset.

than a phrase) and not too long (then it becomes a musical period with cadence etc.). But, of course, there are no rules, the division is and will be subjective."

Nothing about this definition precludes the musical appropriateness of more ambiguous segmentations. In Essen, the primary focus, rather, was to capture a phrase-based level of granularity.

Essen melodies are divided into subsets with different characteristics, such as Polish folk songs and German children's songs, which allows us to investigate algorithmic performance as a function of genre. In our experiments, the subsets specified in Table 1 were used. These sets are interesting because, per set, their melodic content is quite different. The subsets are also fairly large and exhibit characteristic average values for notes per song and notes per phrase. For a given subset, the number of songs reported in this table is smaller than the number available in the original collection because only those songs with simple meters were included.[6] The *All* subset was constructed by merging the other four together.

The Musician Corpus. Ten musical excerpts in very different styles were chosen as the basis of the musician corpus (Table 2).[7] Nineteen musicians with various levels of expertise – including professionals, musicologists, music teachers, and amateurs – were asked to identify salient melodic chunks for each excerpt at the *phrase level* and the *sub-phrase* level. Subphrases were only required to be more fine-grained than phrases, providing additional structure at "the next level down." Instructions were kept deliberately vague. For instance, the term motive was not used because we wanted the musicians to draw upon their musical intuition rather than perform explicit melodic analysis. Subjects were instructed to identify each segment by placing a mark above its first note. In this way, each segment was defined as ending just before the next mark, which ensured that a musician's solution was always a segment stream. For each excerpt, subjects were given a minimal score of the monophonic melody including its meter and a dead-pan MIDI file.

Some excerpts were selected because of their ambiguous musical structure (e.g. Excerpt 5 and 6), others because of their noticeable lack of structure (i.e. Excerpt 7). Essen tunes were included in order to explore how unambiguous a simple folk song might or might not be. Finally, Excerpts 8 through 10 (an improvisation over a Bluegrass standard, a Pop standard, and a Bebop jazz solo) were chosen for practical reasons. When building interactive musical agents, one could imagine people wanting to be able to interact in these kinds of settings.

[6] Meters 4/4, 3/4, 2/4, 3/8, 6/8 were chosen to enable a straightforward automatic generation of the corresponding beat lists.

[7] This data is available at `i11www.ira.uka.de/~musik/segmentation`.

Excerpt	Source	N_b	N_n
1	Essen song *E0356*	17	52
2	Essen song *E0547*	14	39
3	Essen song *Q0034*	12	57
4	Essen song *F0927*	8	45
5	WTC I, Fugue XV (J. S. Bach)	13	90
6	String Quartet op. 76/4, 1st mvt. (J. Haydn)	21	87
7	*Parsival, Karfreitagszauber* (R. Wagner)	11	30
8	*Grey Eagle* (Bluegrass Standard)	8	121
9	*Saving all my love for you* (Goffin & Masser)	11	52
10	*KoKo* (C. Parker)	32	190

Table 2: Properties of the musician corpus. N_b and N_n are the number of bars and notes per excerpt respectively.

4 Experimental Methodology

4.1 Comparing Solutions

The typical way to evaluate an algorithm is to compare its output, $\mathbf{s}$, to some corresponding *reference solution*, $\mathbf{s}^*$. For instance, in Figure 1, this would amount to calculating a *fitness value* that reflects how good LBDM's segmentation vector is with respect to the Essen reference. When considering ambiguity, however, the reference may or may not be "completely right" in a given situation, and several or many segmentations may be equally good or bad. In fact, our musician corpus was motivated by the observation that it seems inadequate to evaluate an algorithm by merely comparing its output to a single "correct" answer. A comparison must also consider the issue of *class inequality*, which arises because the reference solution will typically contain many fewer boundaries than non-boundaries. For example, in the reference segmentations of the *All* subset, on average there are 8.7 non-boundaries per boundary (cf. Table 1).

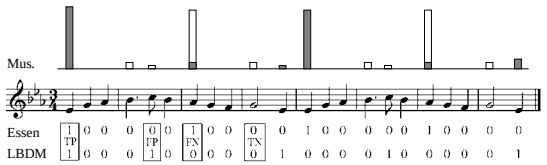


Figure 1: Part of Excerpt 2. Below the score, Essen and LBDM segmentation vectors are displayed. Above, a histogram illustrates the corresponding frequencies with which musicians inserted phrase (gray bars) and sub-phrase (gray and white bars) boundaries between note pairs.

Assuming that each potential segment location can be independently modeled, the difference between two segmentation vectors can be completely described by four simple numbers (Manning and Schütze 2001). The true positives, TP, and true negatives, TN, record

the number of locations in which both vectors contain a value of 1 or 0 respectively. Similarly, the number of false positives, FP, and false negatives, FN, record the number of locations where the algorithm falsely predicts a value of 1 or 0. Each of these types are identified in Figure 1.

A fitness rating for a segmentation can be obtained by combining these numbers into one. However, if one simply counted the proportion of correctly classified locations, i.e. $accuracy = \frac{1}{n}(\mathrm{TP}+\mathrm{TN})$, an algorithm might look good even if it never inserted a segment. This situation arises, because the reference vector will typically contain many more zeros than ones. The *F score* measure eliminates this problem by ignoring TN altogether. It is the harmonic mean of $precision = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$ and $recall = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$:

$$\mathrm{F}(\mathbf{s}, \mathbf{s}^*) = \frac{\mathbf{2 \times precision \times recall}}{\mathbf{precision + recall}}.$$

F score maps onto the range $[0, 1]$, a 1 corresponding to identical sequences and a 0 corresponding to no matches. It balances two conflicting properties: $precision$, which increases as the number of false positives decrease, and $recall$, which increases as the number of false negatives decrease. This is why in Figure 1, F equals .22, a value that lies in between both the $precision$ (.20) and the $recall$ (.25). [8]

An obvious problem with F score is that mismatches are treated in the same way regardless of their location. For example, a segment of length one, which arises when two boundaries are placed right next to each other, might have the same impact on F score as two 1's that are separated by some musically reasonable number of 0's. The underlying issue – a direct result of our independence assumption – is that TP, FP, and FN do not consider a segment's position and length. Another serious problem with F score is that it provides no mention of how to handle ambiguity. Ideally, ambiguity might be handled with a probabilistic model that explicitly considered segment position and length. This idea is further pursued in Spevak, Thom, and Höthker (2002). In our experiments, rather, a segmentation's ambiguity and location dependence are implicitly considered by calculating an average F score. For example, in Exp B and Exp C, a single segmentation $\mathbf{s}^*$ is compared to the musicians' reference segmentations $\mathbf{s_1}, \ldots \mathbf{s_n}$ as follows:

$$\bar{\mathrm{F}} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{F}(\mathbf{s}^*, \mathbf{s_i}).$$

Average F score is appropriately adapted in other settings. For measuring the homogeneity within the musician corpus, the average F score between pairs of musicians' segmentations is computed (Exp A). When evaluating an algorithm's performance on a subset of the Essen collection, the average F score between pairs of computed segmentations and reference segmentations across the subset is used. In this case, the ambiguity of different potential segmentations per melody is not considered, because Essen provides only one reference segmentation per melody.

[8] In this example we have included the initial true positive ($i = 1$) so that TP was not 0. In our experiments, we ignore this value because it contains no useful information, i.e. the first segment always begins at the first note.

4.2 Tuning Parameters

LBDM and Grouper were fit to particular melody sets by varying their adjustable parameters. Suitable parameter sets were searched among a fixed grid of candididates, retaining a setting that produced the best average F score. One important difference between experiments is whether the algorithms were tuned to the Essen or the musician data. Since the Essen subsets are quite large, over-fitting was unlikely to occur. The number of musicians' solutions, on the other hand, was much smaller. Because the musicians' solutions for a given excerpt might not adequately represent all musically reasonable solutions, over-fitting becomes more likely. However this is only an issue when investigating stability, i.e. how well an algorithm with parameters fitted on a training set generalizes on a testing set. When focusing on flexibility, rather, one asks how much a particular algorithm can be fit to a specific situation.

4.3 Experiments

Our quantitative experiments are outlined below:

Exp A: The purpose here was to evaluate the performance of each musician with respect to the others. Per excerpt, average F score was calculated over all pairs of musician solutions. The results are shown in the left-hand portion of Table 3.

Exp B: The purpose of this experiment was to investigate how well an algorithm could perform in principle. Grouper and LBDM were tuned so as to maximize average F score for each excerpt with respect to the musicians' solutions. The best average F scores are reported in the middle portion of Table 3.

Exp C: The purpose here was to investigate how stable the algorithms could be. Each algorithm was configured using the best parameters obtained in Exp D for the *All* subset. Performance was evaluated over each excerpt of the musician corpus. Results are displayed the right portion of Table 3.

Exp D: The purpose of this experiment was to investigate how each algorithm could perform when applied to large, relatively homogeneous sets of melodies. Grouper and LBDM were tuned to all five Essen subsets. The best average F scores obtained for each subset are reported in Table 4.

	Exp A: Ambiguity		Exp B: Optimal Parameters				Exp C: Fixed Parameters			
	Musicians		Grouper		LBDM		Grouper		LBDM	
Exc	Phrase	Sub-phrase	Phrase	Sub-phrase	Phrase	Sub-phrase	Phrase	Sub-phrase	Phrase	Sub-phrase
1	$.70 \pm .27$	$.80 \pm .16$	$.78 \pm .25$	$.88 \pm .15$	$.36 \pm .15$	$.39 \pm .15$	$.27 \pm .17$	$.56 \pm .14$	$.17 \pm .22$	$.29 \pm .16$
2	$.66 \pm .30$	$.80 \pm .31$	$.56 \pm .23$	$.90 \pm .24$	$.14 \pm .16$	$.41 \pm .10$	$.09 \pm .25$	$.04 \pm .14$	$.08 \pm .17$	$.07 \pm .21$
3	$.76 \pm .16$	$.79 \pm .24$	$.82 \pm .14$	$.89 \pm .19$	$.82 \pm .19$	$.89 \pm .19$	$.64 \pm .09$	$.49 \pm .14$	$.81 \pm .15$	$.40 \pm .11$
4	$.57 \pm .36$	$.75 \pm .32$	$.72 \pm .31$	$.77 \pm .22$	$.43 \pm .19$	$.74 \pm .19$	$.72 \pm .31$	$.65 \pm .20$	$.02 \pm .09$	$.01 \pm .05$
5	$.42 \pm .27$	$.50 \pm .23$	$.37 \pm .16$	$.43 \pm .09$	$.39 \pm .20$	$.39 \pm .19$	$.28 \pm .19$	$.34 \pm .10$	$.20 \pm .20$	$.11 \pm .09$
6	$.61 \pm .22$	$.56 \pm .29$	$.48 \pm .11$	$.68 \pm .32$	$.75 \pm .22$	$.66 \pm .33$	$.42 \pm .11$	$.52 \pm .21$	$.46 \pm .18$	$.14 \pm .04$
7	$.14 \pm .25$	$.35 \pm .26$	$.28 \pm .23$	$.54 \pm .29$	$.21 \pm .31$	$.46 \pm .27$	$.16 \pm .23$	$.34 \pm .23$	$.30 \pm .26$	$.45 \pm .25$
8	$.41 \pm .25$	$.36 \pm .19$	$.43 \pm .21$	$.44 \pm .21$	$.39 \pm .28$	$.45 \pm .15$	$.23 \pm .13$	$.28 \pm .15$	$.42 \pm .15$	$.41 \pm .18$
9	$.82 \pm .14$	$.76 \pm .16$	$.89 \pm .12$	$.84 \pm .13$	$.84 \pm .12$	$.73 \pm .11$	$.89 \pm .12$	$.59 \pm .10$	$.65 \pm .13$	$.36 \pm .06$
10	$.82 \pm .12$	$.58 \pm .17$	$.71 \pm .09$	$.64 \pm .13$	$.88 \pm .10$	$.62 \pm .30$	$.67 \pm .08$	$.62 \pm .12$	$.81 \pm .12$	$.52 \pm .19$

Table 3: Qualitative experimental results (average F score $\pm$ standard deviation).

Exp D: Optimal Parameters		
Subset	Grouper	LBDM
Kinder	$.64 \pm .32$	$.51 \pm .30$
Kaszuby	$.70 \pm .35$	$.49 \pm .34$
Ballad	$.60 \pm .36$	$.51 \pm .31$
Fink	$.65 \pm .28$	$.56 \pm .25$
All	$.62 \pm .34$	$.50 \pm .31$

Table 4: Quantitative experimental results (average F score $\pm$ standard deviation).

5 Discussion

How ambiguous is the segmentation task? Exp A shows that the melody excerpts in the musician corpus are ambiguous to a varying extent. Table 3 (left) presents the average inter-expert F score for each excerpt. Results range considerably, from 0.14 for the phrase segmentation of the Wagner melody (Excerpt 7) to 0.82 for *Saving all my love* (Excerpt 9) and *KoKo* (Excerpt 10). Wagner's melodic surface is very smooth, which makes it difficult to segment. The contents of the other two are much less ambiguous. *Saving all my love* has a repetitive structure and rests at verse boundaries. In *KoKo*, the lines are broken up by rests which coincide with phrase boundaries in most musician segmentations. $\bar{\mathrm{F}} = 0.58$ for sub-phrases indicates that finding structure within *KoKo*'s improvised lines was more difficult.

If the musicians disagree on the segmentation of a particular excerpt (a low $\bar{\mathrm{F}}$ in Exp A), an algorithm cannot possibly agree with all of them simultaneously. For this reason, the results obtained in Exp B and C should be interpreted with respect to the baselines reported in Exp A.

How flexible is an algorithm? Exp B provides insight into how flexible each algorithm is (Table 3, middle). Grouper tends to compute segmentations that agree better with the musicians' segmentations than those computed by LBDM (Excerpt 1 and 2), although LBDM outperforms Grouper in some cases (Excerpt 6, phrases). In Excerpt 2, phrase data , LBDM received a low value ($\bar{\mathrm{F}} = 0.14$) even though the musicians agreed on the locations of the phrase boundaries much more often ($\bar{\mathrm{F}} = 0.66$). In this example, the gestalt principle of proximity conflicts with the higher level melodic arch structure. The song starts with two identical arch-shaped phrases (cf. Figure 1), each beginning and ending with the same pitch. There are, therefore, no contour-based cues in the phrase boundaries of this score. Another difficulty is that the longest durations occur in the middle of a phrase, a situation that will confuse any algorithm that relies only on the local features of the melodic surface. Were articulation data also used, perhaps LBDM would have more discriminatory power in this setting. Grouper performs somewhat better on excerpt 2 ($\bar{\mathrm{F}} = 0.56$) because the melody exhibits a regular phrase structure that is captured by the Metrical Parallelism Rule. Furthermore, pitch information is ignored.

At first glance, it might seem puzzling that, in some cases, the sum of $\bar{\mathrm{F}}$ plus or minus its standard deviation is larger than 1 or less than 0. This effect is likely due to a non-symmetric F distribution and to the small number of musician segments that were used to compute the average. In Table 4, which is based on larger data sets, this effect is less noticeable.

How stable is an algorithm? Exp C and D provide insight into how practical each algorithm might be in the case where its parameters cannot be adjusted to a particular situation in advance. For example, consider a real-time performance setting or a situation in which there is not enough relevant hand-segmented data available for parameter tuning. In Exp C, performance was measured for individual corpus excerpts even though the parameters were tuned on the large *All* Essen subset (Table 3, right). As expected, the performance of the algorithms decreases in comparison with Exp B. Interestingly, this decrease is particularly noticeable for sub-phrases (0.26 and 0.30 on average for Grouper and LBDM, respectively). For phrases, it is less significant (0.17 and 0.15), a reasonable explanation being that the phrase granularity in the Essen database might better capture the musicians' notion of phrases than sub-phrases.

In Exp D, Grouper performs better than LBDM, supporting the idea that metrical parallelism has significant influence on the Essen folk songs' phrase structure. However, the standard deviation is high for both algorithms and all subsets. Even with tuned parame-

ters, the minimum F score in a subset is usually zero, meaning that it contains at least one melody for which no boundaries were successfully identified. More to the point, average F score assesses an algorithm's general stability but is not a reliable measure for predicting its behaviour on individual melodies.

Parameters. In general, our tuned LBDM parameters tend to confirm those recommended by Cambouropoulos (e.g. use a pitch and rest weight of $\frac{1}{4}$ and an IOI weight of $\frac{1}{2}$). Notably, the inter-onset stream appears to be the most relevant parameter. Part of the reason for this might be that we used score-data (as opposed to performance data). In Grouper, the length related parameters play an important role: in the best parameter settings, the length weight was never zero. Not surprisingly, in Exp D, the optimal length parameter is correlated with the number of notes per phrase shown in Table 1.

6 Conclusion

In this paper, we reviewed several segmentation stream algorithms and qualitatively highlighted their strengths and weaknesses. Algorithms tend to differ across their temporal processing (local vs. global), the required inputs (score vs. MIDI, real-time vs. batch, harmonic or metric information, training data), their flexibility (hard-wired vs. learned), and their ease-of-use (availability, setup time). For offline applications, Temperley's Grouper provides a well-balanced choice. Although an LBDM-style approach would be ideal in online settings, as is, it performs less well than Grouper. Performance might significantly improve if metric information were included. For such an extension to be of practical use, however, a ready-to-use implementation would be required.

In the remaining sections, we quantified how ambiguous specific segmentation tasks were and used this as a baseline from which to compare Grouper's and LBDM's performance. A notable aspect of our evaluation is that two radically different types of hand-segmented data were used. The first type was made up of large, relatively homogeneous subsets of folk tunes. The second, obtained from a small number of musicians, was composed of relatively distinct segmentations for the same melody. The first type has many stylistically similar songs, but individually, each song had its own distinctive melody and "correct" segmentation. Thus, while an algorithm's user-adjustable parameters can provide the flexibility needed to significantly improve its *average* performance, the variance across songs is impractically large. The second type of dataset, on the other hand, demonstrates that even for "simple" folk songs, ambiguity is substantial, suggesting that measurements made using the first type of dataset should be interpreted cautiously. In particular, when ambiguity is not considered, the obtained measurements might appear overly pessimistic.

Acknowledgments

We are grateful to all those who volunteered to segment the melodic excerpts for us. We also would like to thank R. Bod, E. Cambouropoulos and E. Dahlig for their input.

References

Bod, R. (2001). Memory-based models of melodic analysis: challenging the gestalt principles. *Journal of New Music Research 30*(3).

Cambouropoulos, E. (1998). Musical parallelism and melodic segmentation. In *Proceedings of the XII Colloquium on Musical Informatics*, Gorizia, Italy, pp. 111–114.

Cambouropoulos, E. (2001). The *Local Boundary Detection Model (LBDM)* and its application in the study of expressive timing. In *Proceedings of the International Computer Music Conference (ICMC01)*, Havana, Cuba.

Crawford, T., C. S. Iliopoulos, and R. Raman (1998). String-matching techniques for musical similarity and melodic recognition. In W. B. Hewlett and E. Selfridge-Field (Eds.), *Melodic Similarity: Concepts, Procedures, and Applications*, Chapter 3, pp. 73–100. Cambridge, Mass.: MIT Press.

Dahlig, E. (2002). Personal communication.

Friberg, A., R. Bresin, L. Frydén, and J. Sundberg (1998). Musical punctuation on the microlevel: automatic identification and performance of small melodic units. *Journal of New Music Research 27*(3), 271–292.

Höthker, K., D. Hörnel, and C. Anagnostopoulou (2001). Investigating the influence of representations and algorithms in music classification. *Computers and the Humanities 35*, 65–79.

Lerdahl, F. and R. Jackendoff (1983). *A Generative Theory of Tonal Music*. Cambridge, Mass.: MIT Press.

Manning, C. D. and H. Schütze (2001). *Foundations of Statistical Natural Language Processing*. MIT Press.

Melucci, M., N. Orio, and M. Gambalunga (2000). An evaluation study on music perception for music content-based information retrieval. In *Proceedings of the International Computer Music Conference (ICMC)*, Berlin, pp. 162–165.

Rowe, R. (1993). *Interactive Music Systems: Machine Listening and Composing*. Cambridge, Mass.: MIT Press.

Schaffrath, H. (1997). The essen associative code: A code for folksong analysis. In E. Selfridge-Field (Ed.), *Beyond MIDI: The Handbook of Musical Codes*, Chapter 24, pp. 343–361. Cambridge, Mass.: The MIT Press.

Spevak, C., B. Thom, and K. Höthker (2002). Evaluating melodic segmentation. In *Proceedings of the 2nd Conference on Music and Artificial Intelligence, ICMAI'02*, Edinburgh, Scotland.

Temperley, D. (2001). *The Cognition of Basic Musical Structures*. Cambridge, Mass.: MIT Press.

Thom, B. (2001). *BoB: An Improvisational Music Companion*. PhD thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Appendix A: Selecting Significant Local Maxima

In LBDM, Cambouropoulos (2001) specifies that:

> "...the peaks in the resulting sequence of boundary strengths are taken to be *potential* boundaries." (emphasis ours; page 290)

Because LBDM produces a finite and countable sequence, it is trivial to find all local maxima. This does not mean, however, that LBDM's computational inner workings are thus completely specified. For example, on page 291, Cambouropoulos mentions that adjusting the cutoff threshold for the important peaks can improve LBDM's performance. Unfortunately, from this simple-sounding idea, a large number of algorithmic instantiations could be imagined.

For instance, the threshold could be defined so that its value is context dependent, meaning that its actual value can vary as some function of the melody's overall profile shape. Our implementation falls into this category, introducing a very simple heuristic: define the threshold as some fraction of the overall global maximum. Another context-dependent option would arise if we had defined the threshold so that it resulted in at most some fixed number of peaks crossing it, a scheme that would make LBDM behave a bit more like DOP or Grouper. In order for an algorithm to identify which peaks are important, it needs a bias. As a result, if the profile shapes that an algorithm might see vary considerably, one cannot expect a general-purpose strategy to always perform well. Furthermore, as profiles become more jagged, a robust peak detection strategy becomes all the more important. Unfortunately, in different musical situations, the shapes produced by LBDM can vary significantly (c.f. Appendix 6).

The peak-selection strategy that we implemented is motivated by the following observation: in both our corpus and the Essen collection, segments rarely contains but a single note or two. At the same time, it is not uncommon for LBDM to produce a profile that contains within it the type of contour displayed in the left plot of Figure 6. In this example, it makes perfect sense to insert a boundary between the location of note pair 11 while ignoring the peak at location 13. Provided that the threshold parameter lies below the value at location 12, our threshold will make this distinction. In contrast, the middle and right plots demonstrate some of the weaknesses in our strategy. For example, in the middle plot, it makes sense to select one and only one peak for each of three adjacent regions (between locations 1 through 5, 6 through 11, and 12 through 17). Although the threshold that is displayed identifies the important peak in the middle region, it simultaneously looses its ability to behave reasonably in the other two regions. What is interesting about this example is that adjacent local peaks are divided by exactly two notes — a fairly unlikely segment length. In addition, within the left and right regions, adjacent peaks have relatively similar heights, further complicating which one should be perceived as "more important." The fundamental problem is that a single, vertical threshold is not powerful enough to make these kinds of distinctions. The right plot further demonstrates the difficulty of identifying "important" local peaks. Between locations 1 and 6, the threshold behaves just as we had argued that it should in the leftmost plot. However, when considered with respect to the peak at location 7, location 5 becomes more vital. By extending ones view to cover locations 1 through 17, however, it can be argued that only 11 and 14 (and perhaps 2) are key. At the same time, if we were to lower the threshold to eliminate the boundary at location 7, the only peak in this entire profile that would be identified would be peak 11. The point to take away from this discussion is that local peak-finding is difficult, and the performance of any algorithm that relies on such a procedure strongly depends on the biases that are introduced.

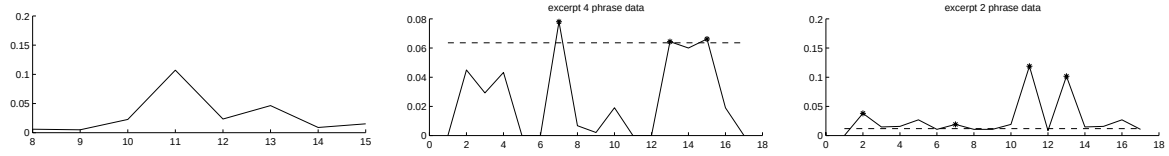


Figure 2: LBDM boundary strength profile examples. The x-axis identifies note pairs and the y-axis identifies boundary strength values. The threshold (dashed line) was chosen for LBDM by tuning to the musicians' data for a given excerpt. Asterisks demonstrate where the boundaries were inserted based on LBDM's profile (solid line).

Appendix B: Tuned Parameters

The parameters that were obtained by tuning the musician corpus (Exp B) and to the Essen subsets (Exp D) are specified in Tables 5 and 6 for LBDM and in Tables 7 and 8 for Grouper respectively.

Excerpt	data	pitch wt	IOI wt	OOI wt	pitch offset	OOI offset	thresh
1	phrase	0.167	0.667	0.167	0.250	0.000	0.250
2	phrase	0.500	0.250	0.250	0.250	0.000	0.100
3	phrase	0.333	0.333	0.333	0.000	0.000	0.250
4	phrase	0.667	0.167	0.167	0.000	0.000	0.500
5	phrase	0.167	0.667	0.167	0.000	0.000	0.160
6	phrase	0.333	0.333	0.333	0.000	0.000	0.130
7	phrase	0.333	0.333	0.333	0.000	0.000	0.280
8	phrase	0.500	0.167	0.333	0.250	0.000	0.550
9	phrase	0.333	0.333	0.333	0.000	0.000	0.130
10	phrase	0.333	0.333	0.333	0.000	0.000	0.130
1	motive	0.167	0.667	0.167	0.250	0.000	0.250
2	motive	0.500	0.333	0.167	0.500	0.000	0.130
3	motive	0.250	0.500	0.250	0.000	0.000	0.160
4	motive	0.667	0.167	0.167	0.000	0.000	0.500
5	motive	0.167	0.500	0.333	0.000	0.000	0.100
6	motive	0.167	0.667	0.167	0.000	0.000	0.050
7	motive	0.250	0.500	0.250	0.000	0.000	0.250
8	motive	0.333	0.333	0.333	0.000	0.000	0.190
9	motive	0.250	0.500	0.250	0.000	0.000	0.100
10	motive	0.333	0.333	0.333	0.000	0.000	0.130

Table 5: LBDM tuned parameters in Exp B.

essen subset	pitch wt	IOI wt	OOI wt	pitch offset	OOI offset	thresh
All	0.167	0.333	0.500	0.500	0.000	0.400
Ballad	0.167	0.500	0.333	0.250	0.000	0.450
Fink	0.167	0.333	0.500	0.000	0.000	0.280
Kaszuby	0.167	0.333	0.500	0.000	0.250	0.450
Kinder	0.167	0.667	0.167	0.000	0.000	0.220

Table 6: LBDM tuned parameters in Exp D.

Excerpt	data	optimal length	length weight	gap weight	phase3 penalty	phase4 penalty	average fscore ± std. deviation
1	phrase	12	300	100	100	100	0.779 ±0.248
2	phrase	2	100	100	500	300	0.561 ±0.229
3	phrase	14	500	300	100	100	0.817 ±0.143
4	phrase	8	300	100	300	300	0.723 ±0.312
5	phrase	2	100	500	700	300	0.374 ±0.163
6	phrase	14	500	300	100	100	0.478 ±0.111
7	phrase	8	300	900	300	100	0.285 ±0.226
8	phrase	12	900	100	700	700	0.429 ±0.205
9	phrase	10	100	100	100	100	0.893 ±0.121
10	phrase	10	700	500	100	100	0.707 ±0.087
1	motive	4	300	100	300	300	0.876 ±0.154
2	motive	2	100	100	500	300	0.895 ±0.240
3	motive	6	500	700	300	100	0.886 ±0.187
4	motive	4	300	100	500	100	0.766 ±0.222
5	motive	10	100	900	500	100	0.431 ±0.088
6	motive	6	300	100	100	100	0.684 ±0.316
7	motive	4	300	300	100	300	0.544 ±0.293
8	motive	12	900	100	700	700	0.436 ±0.213
9	motive	6	100	300	100	300	0.841 ±0.126
10	motive	12	900	500	300	300	0.642 ±0.131

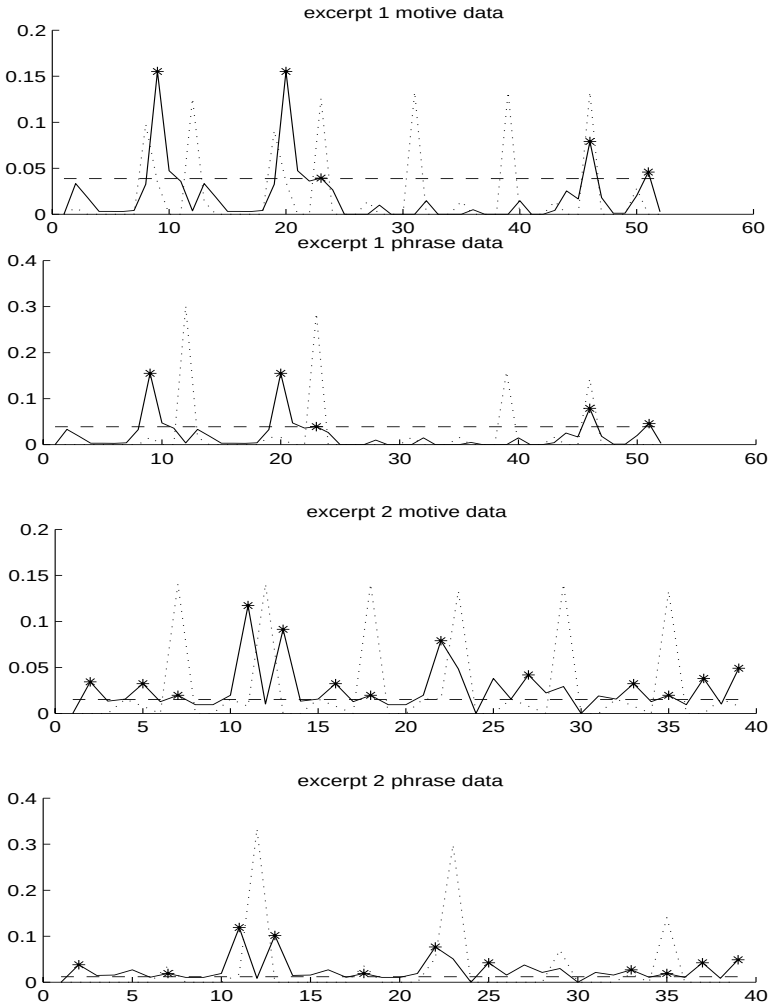
Table 7: Grouper tuned parameters in Exp B.

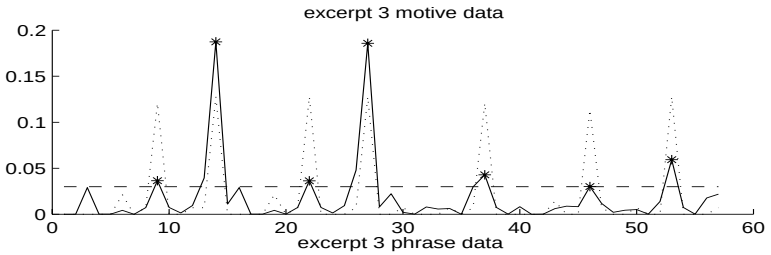
essen subset	optimal length	length weight	gap weight	phase3 penalty	phase4 penalty	average fscore ± std. deviation
All	10	700	400	300	600	0.616 ± 0.339
Ballad	10	700	600	300	300	0.595 ± 0.358
Fink	10	500	500	300	300	0.645 ± 0.282
Kaszuby	11	700	400	600	600	0.697 ± 0.351
Kinder	8	600	600	600	300	0.636 ± 0.323

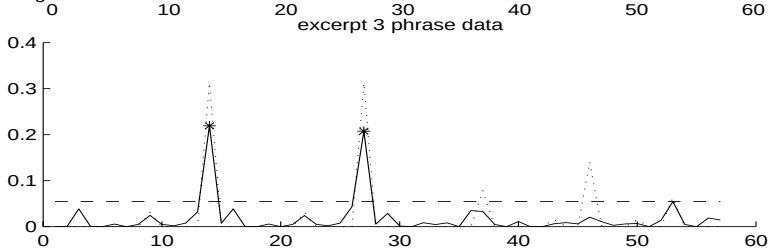
Table 8: Grouper tuned parameters in Exp D.

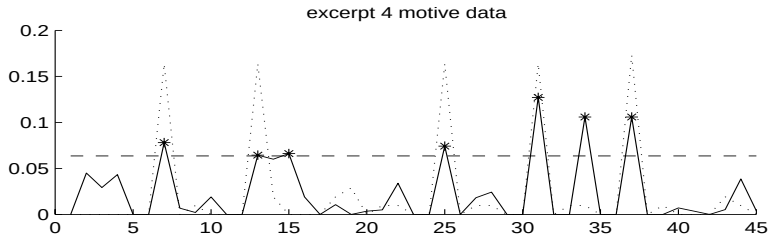
Appendix C: LBDM Profiles for Exp B

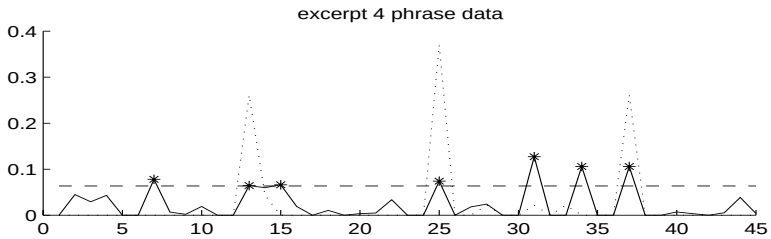
This appendix presents the LBDM boundary strength profiles generated using each parameter sets outlined in Table 5. Per excerpt, two plots are given, one displaying LBDM's behavior when it was tuned on the phrase-based data and another when tuned on motive-based data. Dashed lines indicate the threshold value that was chosen when tuning LBDM, solid lines identify LBDM's boundary strength profile, and asterisks identify which boundaries were selected from this profile. The dotted line, on the other hand, illustrates the corresponding frequencies with which musicians inserted their boundaries. Given that the musician data was used to tune LBDM's parameters, it might seem surprising that the dotted and solid lines are often not aligned on the same locations. This situation is explained by the fact that, at times, the structure that is driving the melodic segmentation process cannot be teased out from a mere local view of a melody's surface.

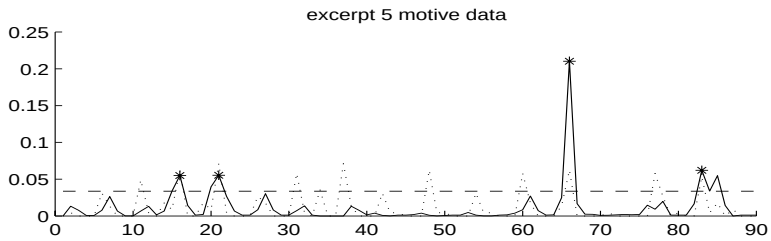

excerpt 1 motive data
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
excerpt 1 phrase data
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
excerpt 2 motive data
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40
excerpt 2 phrase data
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40

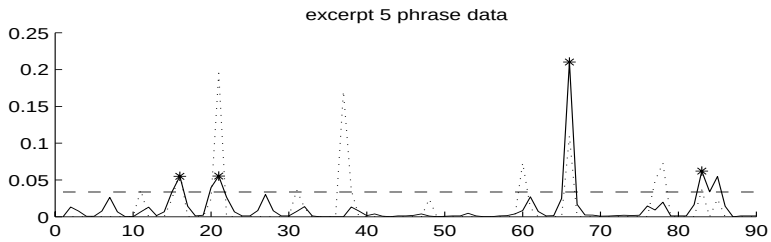

excerpt 3 motive data
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60

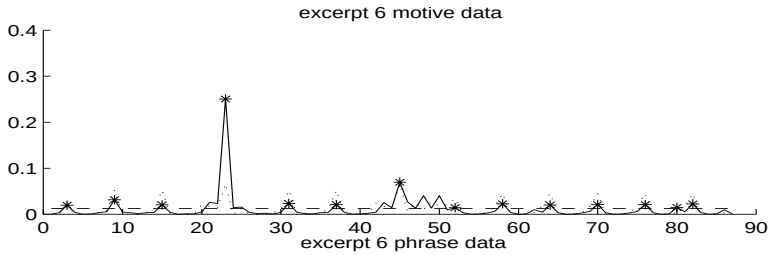

excerpt 3 phrase data
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60

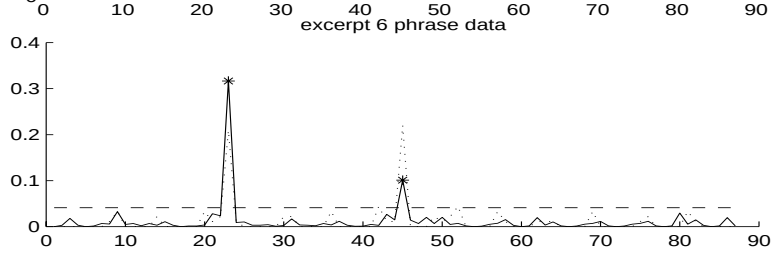

excerpt 4 motive data
0.2
0.15
0.1
0.05
0
0
5
10
15
20
25
30
35
40
45

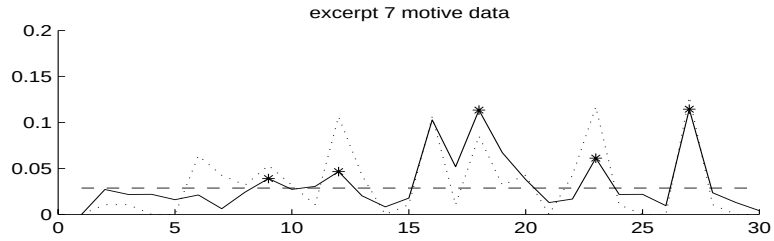

excerpt 4 phrase data
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30
35
40
45

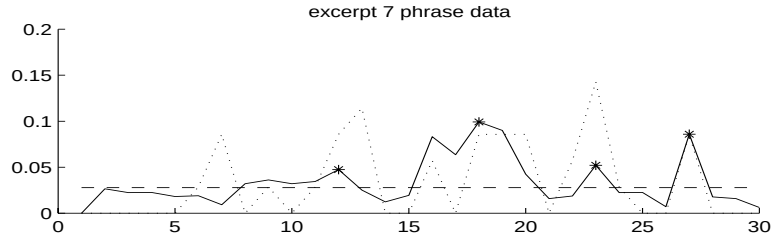

excerpt 5 motive data
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
70
80
90

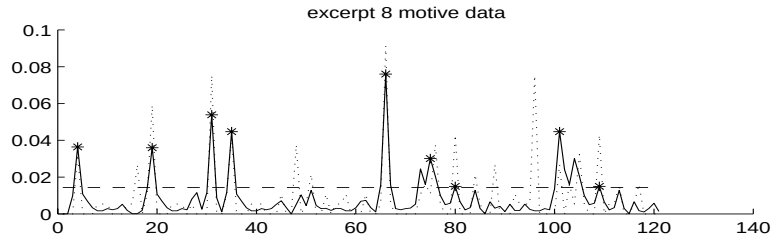

excerpt 5 phrase data
0.25
0.2
0.15
0.1
0.05
0
0
10
20
30
40
50
60
70
80
90

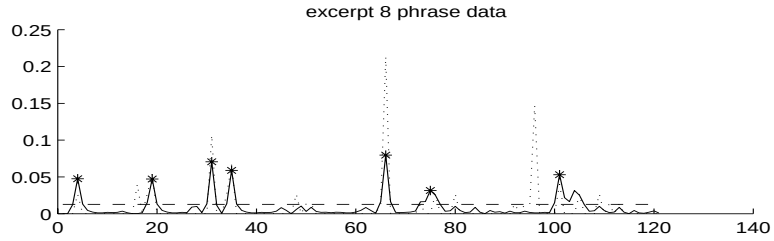

excerpt 6 motive data
0.4
0.3
0.2
0.1
0
0 10 20 30 40 50 60 70 80 90

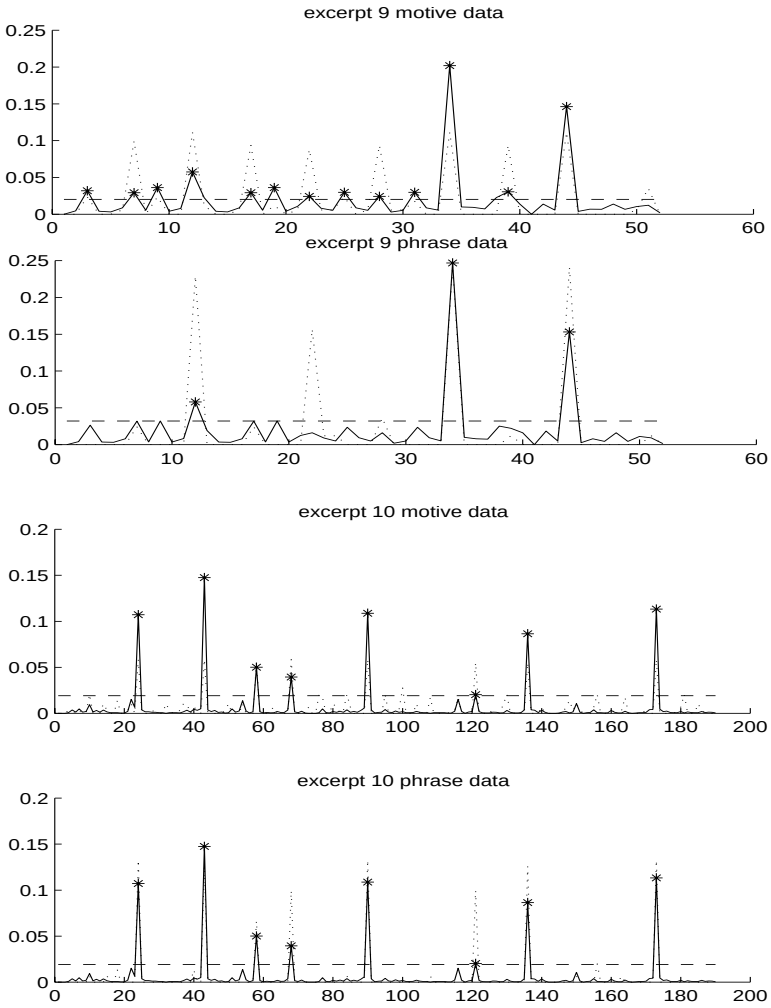

excerpt 6 phrase data
0.4
0.3
0.2
0.1
0
0 10 20 30 40 50 60 70 80 90


excerpt 7 motive data
0.2
0.15
0.1
0.05
0
0 5 10 15 20 25 30


excerpt 7 phrase data
0.2
0.15
0.1
0.05
0
0 5 10 15 20 25 30


excerpt 8 motive data
0.1
0.08
0.06
0.04
0.02
0
0 20 40 60 80 100 120 140


excerpt 8 phrase data
0.25
0.2
0.15
0.1
0.05
0
0 20 40 60 80 100 120 140


excerpt 9 motive data
excerpt 9 phrase data
excerpt 10 motive data
excerpt 10 phrase data

Appendix D: Probabilistic Interpretations of Segmentation Solutions

When musicians segment the same note list input into different output solutions, the mapping – no longer a mathematical function – can be stochastically described. It makes sense to consider modeling a solution's goodness – i.e. its *fitness* – from within the a probabilistic framework. Ideally, we would develop a model that, given a musically plausible segmentation, returns a higher likelihood than when given a less plausible solution as input. Although our population of musician solutions provides an excellent base from which to configure such a model, two difficulties arise in using this data. First, the class inequality between 1's and 0's makes it hard to build a model that is not overwhelmed by the number of 0's. Second, because musicians tend to structure the locations of their 1's in complex ways, the assumption that individual boundary locations can be independently modeled breaks down.

The first issue is best appreciated by evaluating some musician data. Consider the data collected for Excerpt 1 (Figure 3). A simple but naive approach would combine this data into a probabilistic model by normalizing over the total number of counts, resulting in $\vec{w}$ (Figure 4, top). This distribution could be used to weight each 1 in a solution as:

$$fit \ \propto \ \Pi_{i=1}^{n} {w_i}^{s_i}.^{9}$$

This fitness has the desirable property that it will be larger for those solutions that have their 1's aligned in locations that the musicians preferred. A big problem with this weighting scheme is that, for each 1, another multiplicative decreasing factor is introduced. As a result, solutions are penalized according to the number of segments they contain. One way to decouple fitness from the total number of 1's would be to also weigh each 0 in a solution. For example, consider the following fitness:

$$fit \ \propto \ \Pi_{i=1}^{n} {w_i}^{s_i} {v_i}^{(1-s_i)},$$

where $v_i = 1 - w_i$. Unfortunately, the large $\vec{v}$ values that result (Figure 4 middle) mean that this scheme still prefers 0's to 1's. Normalizing over $\vec{v}$ (Figure 4, bottom),

$$v_i' = \frac{v_i}{\Sigma_{i=1}^{n} v_i},$$

would solve this problem, producing higher fitnesses for those solutions whose boundaries coincide with peaks in $\vec{w}$. The unfortunate aspect of this normalization is that it deemphasizes the peaks in $\vec{v'}$ to the point where apt distinguishing between common versus uncommon 0 locations cannot be made.

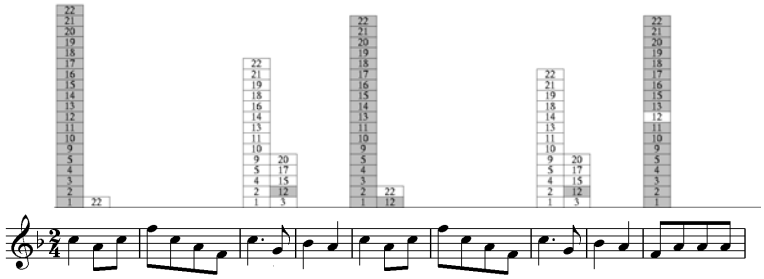


Figure 3: Part of the sub-phrase data collected for Excerpt 1. The histogram illustrates the corresponding frequencies with which musicians inserted phrase (gray bars) and sub-phrase (gray and white bars) boundaries. The number enclosing each count identifies which musician inserted it.

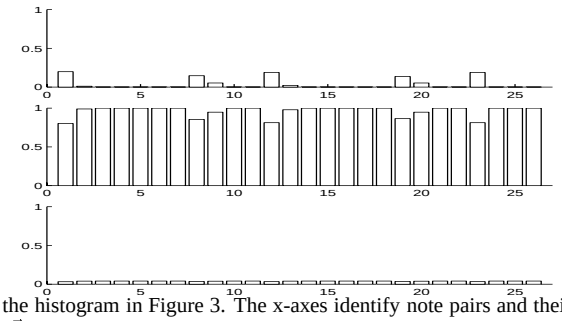


Figure 4: Three transformations of the histogram in Figure 3. The x-axes identify note pairs and their corresponding y-axes identify various weights – top: $\vec{w}$; middle: $\vec{v}$; bottom: $\vec{v'}$.

A closer look at Figure 3 also demonstrates why the complex, underlying structure of the musicians solutions invalidates the use of a simple histogram-based weighting scheme. The product in the preceding fitness schemes all rely on the assumption that different locations i and i' can be treated independently. For example consider the relative preference (14 to 5) musicians had for inserting boundaries between note pairs 8 and 9. Although no musician placed two counts right next to each other, with our independence assumption, a solution that had a count in both location 8 and 9 would look better than a solution that had a count in 8 and 13. Nonetheless, musical segments of length one are not very common.

[9] The '$\propto$' indicates that fit must still be normalized if it is to have a probabilistic interpretation.